软件工程师开发大系

Java Web 开发实例大全（基础卷）

600个典型实例及源码分析，涵盖23个应用方向

工作应用**速查** 项目开发**参考** 学习实战**练习**

软件开发技术联盟◎编著

- **工作应用速查：** 本书实例全面、系统，涉及程序开发的各个方面，适合各级程序开发人员速查速用。
- **项目开发参考：** 程序员借助本书提供的实例源代码，可以快速搭建工程项目，提高开发效率。
- **学习实战练习：** 入门者的实战训练大全，不但可以激发学习兴趣，更可提高编程实战能力和编程思维水平。

DVD

清华大学出版社

VISIT…

LANZAROTE Caliente.com

软件工程师开发大系

Java Web 开发实例大全

（基础卷）

软件开发技术联盟　编著

清华大学出版社

北　京

内 容 简 介

《Java Web 开发实例大全（基础卷）》筛选、汇集了 Java Web 开发从基础知识到高级应用各个层面约 600 个实例及源代码，每个实例按实例说明、关键技术、设计过程、详尽注释、秘笈心法的顺序进行了分析解读。全书分为 6 篇 23 章，主要内容有开发环境搭建、Java 语言基础、HTML/CSS 技术、JSP 基础与内置对象、JavaBean 技术、Servlet 技术、过滤器与监听器技术、JSTL 标签库、JavaScript 技术、Ajax 技术、文件基本操作及文件上传下载、文件的批量管理、图像生成、图像操作、多媒体应用、窗口的应用、导航条的应用、表单的应用、表格的操作、JSP 操作 Word、JSP 操作 Excel、报表与打印、综合应用等。配书光盘附有实例源代码及部分讲解视频。

《Java Web 开发实例大全（基础卷）》既适合 Java Web 程序员参考和查阅，也适合 Java Web 初学者，如高校学生、软件开发培训学员及相关求职人员学习、练习、速查使用。

本书封面贴有清华大学出版社防伪标签，无标签者不得销售。
版权所有，侵权必究。侵权举报电话：010-62782989　13701121933

图书在版编目（CIP）数据

Java Web 开发实例大全．基础卷/软件开发技术联盟编著．—北京：清华大学出版社，2016
（软件工程师开发大系）
ISBN 978-7-302-39952-0

I. ①J…　II. ①软…　III. ①JAVA 语言−程序设计　IV. ①TP312

中国版本图书馆 CIP 数据核字（2015）第 085930 号

责任编辑：赵洛育
封面设计：李志伟
版式设计：魏　远
责任校对：王　云
责任印制：

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编：**100084
社 总 机：010-62770175　　**邮　　购：**010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：203mm×260mm　**印　　张：**58　**字　　数：**1909 千字
（附光盘 1 张）
版　　次：2016 年 1 月第 1 版　**印　　次：**2016 年 1 月第 1 次印刷
印　　数：1～3500
定　　价：128.00 元

产品编号：052266-01

前　言

Preface

特别说明：

《Java Web 开发实例大全》分为基础卷（即本书）和提高卷两册。本书的前身是《Java Web 开发实战 1200 例（第 I 卷）》。

编写目的

1．方便程序员查阅

程序开发是一项艰辛的工作，挑灯夜战、加班加点是常有的事。在开发过程中，一个技术问题可能会占用几天甚至更长时间。如果有一本开发实例大全可供翻阅，从中找到相似的实例作参考，也许几分钟就可以解决问题。本书编写的主要目的就是方便程序员查阅、提高开发效率。

2．通过分析大量源代码，达到快速学习之目的

本书提供了约 600 个开发实例及源代码，附有相应的注释、实例说明、关键技术、设计过程和秘笈心法，对实例中的源代码进行了比较透彻的解析。相信这种办法对激发学习情趣、提高学习效率极有帮助。

3．通过阅读大量源代码，达到提高熟练度之目的

俗话说“熟能生巧”，读者只有通过阅读、分析大量源代码，并亲自动手去做，才能够深刻理解、运用自如，进而提高编程熟练度，适应工作之需要。

4．实例源程序可以“拿来”就用，提高了效率

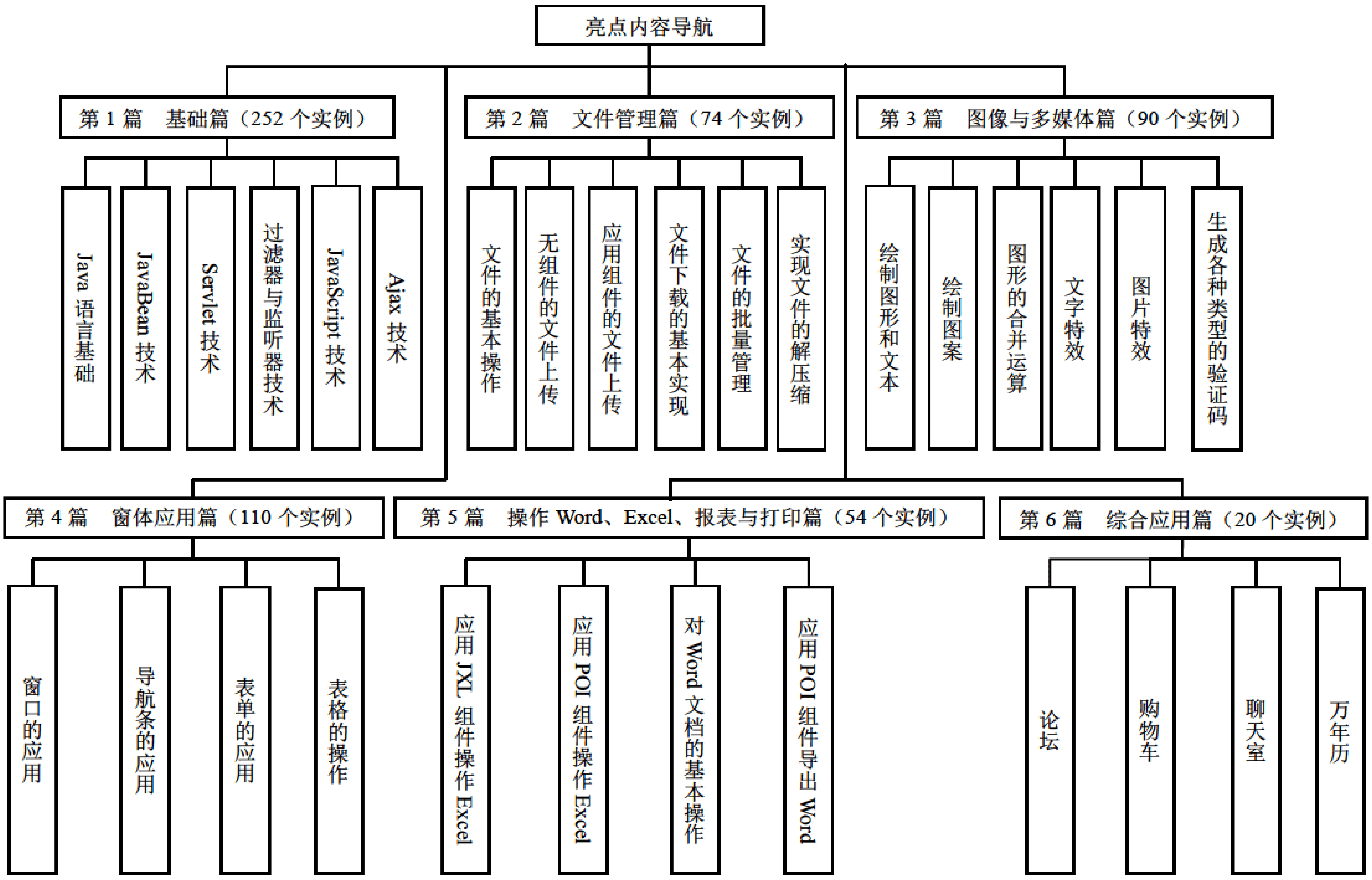
本书的很多实例，可以根据实际应用需求稍加改动，拿来就用，不必再去从头编写，从而节约了时间，提高了工作效率。

本书内容

全书分为 6 篇 23 章，主要内容有开发环境搭建、Java 语言基础、HTML/CSS 技术、JSP 基础与内置对象、JavaBean 技术、Servlet 技术、过滤器与监听器技术、JSTL 标签库、JavaScript 技术、Ajax 技术、文件基本操作及文件上传下载、文件的批量管理、图像生成、图像操作、多媒体应用、窗口的应用、导航条的应用、表单的应用、表格的操作、JSP 操作 Word、JSP 操作 Excel、报表与打印、综合应用等。

书中所选实例均来源于一线开发人员的项目开发实践，囊括了开发中经常碰到和需要解决的热点、难点问题，使读者可以快速解决开发中的难题，提高编程效率。本书知识结构如下图所示。

本书在讲解实例时采用统一的编排样式，多数实例由“实例说明”“关键技术”“设计过程”“秘笈心法”4 部分构成。其中，“实例说明”部分采用图文结合的方式介绍实例的功能和运行效果；“关键技术”部分介绍了实例使用的重点、难点技术；“设计过程”部分讲解了实例的详细开发过程；“秘笈心法”部分给出了与实例相关的技巧和经验总结。



本书特点

1. 实例极为丰富

本书精选了约 600 个实例，另外一册《Java Web 开发实例大全（提高卷）》也精选了提高部分约 600 个实例，这样，两册图书总计约 1200 个实例，可以说是目前市场上实例最多、知识点最全面、内容最丰富的软件开发类图书，涵盖了编程中各个方面的应用。

2. 程序解释详尽

本书提供的实例及源代码，附有相应的注释、实例说明、关键技术、设计过程和秘笈心法。分析解释详尽，便于快速学习。

3. 实践实战性强

本书的实例及源代码很多来自现实开发中，光盘中给出了绝大多数实例的全部源代码，读者可以直接调用、研读、练习。

关于光盘

1. 实例学习注意事项

读者在按照本书学习、练习的过程中，可以从光盘中复制源代码，修改时注意去掉源码文件的只读属性。**有些实例需要使用相应的数据库或第三方资源，在使用前需要进行相应配置，具体步骤请参考书中或者光盘中的配置说明。**

2．实例源代码及视频位置

本书光盘提供了实例的源代码，位置在光盘中的“MR\章号\实例序号”文件夹下，例如，“MR\04\096”表示实例 096，位于第 4 章。部分实例提供的视频讲解，也可根据以上方式查找。**由于有些实例源代码较长，限于篇幅，图书中只给出了关键代码，完整代码放置在光盘中。**

3．视频使用说明

本书提供了部分实例的视频讲解，在目录中标题前边有视频图标的实例，即表示在光盘中有视频讲解。视频采用 EXE 文件格式，无须使用播放器，双击就可以直接播放。

读者对象

Java Web 程序员，Java Web 初学者，如高校大学生、求职人员、培训机构学员等。

本书服务

如果您使用本书的过程中遇到问题，可以通过如下方式与我们联系。

☑ 服务 QQ：4006751066

☑ 服务网站：http://www.mingribook.com

本书作者

本书由软件开发技术联盟组织编写，具体参与编写的程序员有赛奎春、王小科、王国辉、王占龙、高春艳、张鑫、杨丽、辛洪郁、周佳星、申小琦、张宝华、葛忠月、王雪、李贺、吕艳妃、王喜平、张领、杨贵发、李根福、刘志铭、宋禹蒙、刘丽艳、刘莉莉、王雨竹、刘红艳、隋光宇、郭鑫、崔佳音、张金辉、王敬洁、宋晶、刘佳、陈英、张磊、张世辉、高茹、陈威、张彦国、高飞、李严。本书出版方面的项目负责人刘利民编辑和杨静华编辑以及其他编校人员为本书的出版付出了努力，在此一并致谢！

编　者

目　录

Contents

第 1 篇　基础篇

第 1 章　开发环境搭建 2

1.1　JDK 开发工具包 3

实例 001　JDK 的下载 3
实例 002　JDK 的安装 5
实例 003　设置 Java 环境变量 7
实例 004　使用命令行工具测试 JDK 8
实例 005　在命令行编译 Java 源码 8

1.2　Tomcat 服务器 9

实例 006　下载 Tomcat 服务器 9
实例 007　安装 Tomcat 服务器 10
实例 008　启动 Tomcat 并测试 11
实例 009　通过 Eclipse 部署与发布 Web 应用 12
实例 010　修改 Tomcat 服务器的端口号 14
实例 011　配置 Tomcat 的虚拟主机 15
实例 012　在 Tomcat 下如何手动部署 Web 应用 15
实例 013　Tomcat 如何制定主机访问 16
实例 014　Tomcat 如何添加管理员 16
实例 015　Tomcat 常用的优化技巧 17

1.3　Linux 系统配置 JDK 与 Tomcat 服务器 18

实例 016　在 Linux 系统下安装配置 JDK 18
实例 017　在 Linux 系统下安装配置 Tomcat 20

第 2 章　Java 语言基础 22

2.1　基本语法 23

实例 018　输出错误信息与调试信息 23
实例 019　从控制台接收输入字符 23
实例 020　重定向输出流实现程序日志 24
实例 021　自动类型转换与强制类型转换 25

2.2　运算符 26

实例 022　加密可以这样简单（位运算） 26
实例 023　用三元运算符判断奇数和偶数 27
实例 024　更精确地使用浮点数 28
实例 025　不用乘法运算符实现 2×16 29
实例 026　实现两个变量的互换（不借助第 3 个变量） 30

2.3　条件语句 31

实例 027　判断某一年是否为闰年 31
实例 028　验证登录信息的合法性 32
实例 029　为新员工分配部门 32
实例 030　用 switch 语句根据消费金额计算折扣 33
实例 031　判断用户输入月份的季节 34

2.4　循环控制 35

实例 032　使用 while 与自增运算符循环遍历数组 35
实例 033　使用 for 循环输出杨辉三角 36
实例 034　使用嵌套循环在控制台上输出九九乘法表 37
实例 035　用 while 循环计算 1+1/2!+1/3!…1/20! 38
实例 036　用 for 循环输出空心的菱形 39
实例 037　foreach 循环优于 for 循环 40
实例 038　终止循环体 41
实例 039　循环体的过滤器 42
实例 040　循环的极限 42

2.5　常用排序 43

实例 041　冒泡排序法 43
实例 042　快速排序法 44
实例 043　选择排序法 45
实例 044　插入排序法 46
实例 045　归并排序法 47

2.6　算法应用 48

实例 046　算法应用——百钱买百鸡 48
实例 047　算法应用——韩信点兵 49
实例 048　算法应用——斐波那契数列 49
实例 049　算法应用——水仙花数 50
实例 050　算法应用——素数 51
实例 051　算法应用——汉诺塔 52

第 3 章 HTML/CSS 技术......53
3.1 页面效果......54
实例 052 统一站内网页风格......54
实例 053 设置超链接文字的样式......57
实例 054 网页换肤......58
实例 055 滚动文字......59
实例 056 制作渐变背景......61
实例 057 CSS 控制绝对定位......62
实例 058 CSS 控制垂直居中......62
实例 059 CSS 实现的图文混排......63
3.2 表格样式......64
实例 060 只有外边框的表格......64
实例 061 彩色外边框的表格......65
实例 062 单元格的边框变色......66
实例 063 表格外边框具有霓虹灯效果......67
实例 064 控制表格指定外边框不显示......68
实例 065 背景颜色渐变的表格......69
实例 066 表格隔行变色......70
实例 067 表格隔列变色......71
实例 068 鼠标经过表格时，显示提示信息......72
3.3 鼠标样式......72
实例 069 显示自定义的鼠标形状......73
实例 070 动画光标......74
3.4 文字及列表样式......75
实例 071 应用删除线样式标记商品特价......75
实例 072 在文字上方标注说明标记......76
实例 073 改变首行文字的样式......76
实例 074 使文字具有下划线效果......77
实例 075 指定图标的列表项......78
3.5 文字特效......79
实例 076 文字的发光效果......79
实例 077 文字的阴影效果......80
实例 078 文字的渐变阴影效果......81
实例 079 文字的图案填充效果......81
实例 080 文字的探照灯效果......82
实例 081 文字的闪烁效果......83
实例 082 文字的空心效果......84
实例 083 文字的浮雕效果......85
实例 084 文字的阳文效果......86
实例 085 文字的雪雕效果......87
实例 086 文字的火焰效果......87
实例 087 文字的扭曲动画......88
实例 088 输出文字......89
3.6 图片滤镜特效......90
实例 089 图片的半透明效果......90
实例 090 图片的模糊效果......91
实例 091 图片的渐隐渐现效果......92
实例 092 图片的水波纹效果......93
实例 093 图片的灰度效果......94
实例 094 图片的动态说明文字......94

第 4 章 JSP 基础与内置对象......96
4.1 JSP 的基本应用......97
实例 095 自定义错误页面......97
实例 096 导入版权信息......98
实例 097 应用 Java 程序片段动态生成表格......99
实例 098 应用 Java 程序片段动态生成下拉列表......100
实例 099 同一页面中的多表单提交......101
实例 100 在 JSP 脚本中插入 JavaScript 代码......102
实例 101 将页面转发到用户登录页面......103
4.2 JSP 内置对象......105
实例 102 获取表单提交的信息......105
实例 103 获取访问请求参数......107
实例 104 将表单请求提交到本页......108
实例 105 通过 request 对象进行数据传递......109
实例 106 通过 cookie 保存并读取用户登录信息......110
实例 107 实现重定向页面......112
实例 108 防止表单在网站外部提交......113
实例 109 通过 Application 对象实现网站计数器......114
实例 110 记录用户 IP 地址的计数器......115
实例 111 只对新用户计数的计数器......118
实例 112 统计用户在某一页停留的时间......120
实例 113 应用 session 对象实现用户登录......121
实例 114 统计用户在站点停留的时间......123
实例 115 判断用户是否在线......126
实例 116 实时统计在线人数......129
4.3 JSP 的自定义标签......130
实例 117 带标签体的自定义标签......130
实例 118 自定义多次执行的循环标签......131
实例 119 自定义显示版权信息标签......133
实例 120 自定义图片浏览标签......134
实例 121 自定义文件下载的标签......136
实例 122 自定义数据查询的标签......138
实例 123 自定义生成随机数的标签......140

实例 124　自定义生成系统菜单的标签 142

第 5 章　JavaBean 技术 145

5.1　字符串处理 146

实例 125　小写金额转换成大写金额 146
实例 126　转换输入文本中的回车和空格 148
实例 127　计算字符串的实际长度 150
实例 128　字符串截取 151
实例 129　字符串转换成数组 152
实例 130　数组转换为字符串 154
实例 131　将整型值转换为字符串 155
实例 132　将字符串型转换为整型 157
实例 133　把整型数据格式化为指定长度的字符串 158
实例 134　将长整型的数字分位显示 160
实例 135　过滤输入字符串中的危险字符 162
实例 136　过滤字符串中的空格与 NULL 值 163
实例 137　获得汉字的拼音简码 165

5.2　数据验证 167

实例 138　判断字符串是否以指定字符开头 167
实例 139　检查字符串是否包含英文字母 168
实例 140　检查字符串是否包含数字 170
实例 141　判断用户输入的日期是否为当前日期 171
实例 142　判断是否为数字 173
实例 143　判断用户名是否有效 175

5.3　日期时间处理 176

实例 144　将指定日期字符串转换为 Calendar 对象 176
实例 145　将 Calendar 对象转换为日期时间字符串 178
实例 146　获得系统当前时间的字符串格式 179
实例 147　计算出两个日期相差的天数 181

5.4　输出实用的 HTML 代码 182

实例 148　输出提示信息的方法 182
实例 149　输出分页导航的方法 184
实例 150　版权信息的生成方法 185

5.5　窗口与对话框 186

实例 151　弹出提示对话框并重定向网页 186
实例 152　打开指定大小的新窗口 187

5.6　对数据库操作的 JavaBean 189

实例 153　连接数据库的方法 189
实例 154　数据库查询的方法 190
实例 155　带参数的数据查询 192
实例 156　向数据表中插入数据的方法 194
实例 157　数据修改的方法 196
实例 158　数据删除的方法 199
实例 159　数据分页的方法 201
实例 160　对结果集进行分页的方法 203
实例 161　关闭数据库的方法 205
实例 162　数据库事务处理的方法 206
实例 163　调用数据库存储过程的方法 208

第 6 章　Servlet 技术 211

6.1　Servlet 基础 212

实例 164　动态生成 HTML 文档 212
实例 165　在 Servlet 中实现页面转发 213
实例 166　在 Servlet 中实现页面重定向 214
实例 167　在 Servlet 中处理表单提交的数据 215
实例 168　在 Servlet 中向客户端写 Cookie 信息 217
实例 169　在 Servlet 中将 JavaBean 对象传递到 JSP 页 218
实例 170　在 Servlet 中获取 Web 路径和文件真实路径 220
实例 171　在 Servlet 中访问 Web 应用的工作目录 221

6.2　Servlet 应用 223

实例 172　记录用户访问次数 223
实例 173　将数据导出到 Excel 224
实例 174　利用 Servlet 生成动态验证码 226
实例 175　避免客户端访问的并发问题 228
实例 176　在 Servlet 中使用 JDBC 访问数据库 229
实例 177　利用 Servlet 访问数据库连接池 232
实例 178　Servlet 实现的个人所得税计算器 233
实例 179　利用 Servlet 实现用户永久登录 235

第 7 章　过滤器与监听器技术 239

7.1　Servlet 过滤器 240

实例 180　创建过滤器 240
实例 181　防盗链过滤器 241
实例 182　日志记录过滤器 242
实例 183　字符替换过滤器 244
实例 184　异常捕获过滤器 245
实例 185　验证用户身份 Filter 过滤器 247
实例 186　字符编码过滤器 248
实例 187　使用过滤器监控网站流量 250
实例 188　防止页面缓存的过滤器 251
实例 189　通过过滤器控制页面输出内容 253
实例 190　使用过滤器自动生成静态页面 256

实例 191　文件上传过滤器............258
实例 192　权限验证过滤器............260
7.2　监听器的应用............262
实例 193　监听在线用户............262
实例 194　应用监听器使服务器端免登录............264

第 8 章　JSTL 标签库............267
8.1　JSTL Core 标签库............268
实例 195　利用 JSTL 标签实现网站计数器............268
实例 196　根据参数请求显示到不同的页面............269
实例 197　利用<c:forTokens>标签遍历字符串............270
实例 198　利用 JSTL 选取随机数给予不同的提示信息............271
实例 199　利用<c:forEach>标签遍历 List 集合的元素............272
实例 200　利用 JSTL 标签导入用户注册协议............273
8.2　JSTL I18N 标签库............275
实例 201　利用 JSTL 标签设置请求的字符编码............275
实例 202　利用 JSTL 标签实现国际化............276
实例 203　利用<fmt:setLocale>显示所有地区的数据格式............277
实例 204　利用<fmt:timeZone>显示不同地区的时间............279
实例 205　利用<fmt:formatDate>标签对日期格式化............280

第 9 章　JavaScript 技术............283
9.1　数据验证............284
实例 206　通过正则表达式验证日期............284
实例 207　验证输入的日期是否正确............285
实例 208　检查表单元素的值是否为空............287
实例 209　验证是否为数字............288
实例 210　验证 E-mail 是否正确............290
实例 211　验证电话号码是否正确............292
实例 212　验证手机号码是否正确............293
实例 213　验证字符串是否为汉字............294
实例 214　验证身份证号码是否有效............295
实例 215　验证车牌号码是否有效............296
实例 216　验证网站地址是否有效............298
实例 217　验证数量和金额............299
实例 218　验证字符串是否以指定字符开头............300
实例 219　限制输入字符串的长度............301
实例 220　验证输入字符串是否包含特殊字符............302
实例 221　限制用户不允许输入中文字符............303
9.2　字符串处理............304
实例 222　小写金额转换为大写金额............304
实例 223　去掉字符串左右空格............306
实例 224　将数字字符串格式化为指定长度............307
实例 225　限制 Textarea 文本域内容的长度............308
实例 226　将长数字分位显示............309
实例 227　将 RGB 格式的颜色值转换为十六进制............311
实例 228　从指定 URL 中提取文件名............312
9.3　日期时间处理............313
实例 229　计算两个日期相差的天数............314
实例 230　计算两个日期相差的小时数............316
实例 231　计算某一天是星期几............317
实例 232　显示长日期格式的系统时间............318
实例 233　实时显示系统时间............319
实例 234　倒计时............320
9.4　使用 JavaScript 控制 DOM............321
实例 235　创建节点............321
实例 236　添加节点............322
实例 237　为下拉列表增加选项............323
实例 238　删除下拉列表的选项............324
实例 239　可编辑表格............324

第 10 章　Ajax 技术............326
10.1　定时业务............327
实例 240　考试计时并自动提交试卷............327
实例 241　自动保存草稿............331
10.2　改善用户体验............332
实例 242　检查用户名是否重复............333
实例 243　验证用户登录............335
实例 244　限时竞拍............337
实例 245　带进度条的文件上传............342
实例 246　仿 Google Suggest 自动完成............345
实例 247　实现无刷新分页............348
实例 248　实时弹出气泡提示窗口............352
10.3　动态加载数据............356
实例 249　实时显示最新商品及报价............356
实例 250　实时显示聊天内容............359
实例 251　实现快速浏览............361
实例 252　动态多级联下拉列表............363

第 2 篇　文件管理篇

第 11 章　文件基本操作及文件上传下载 372

11.1　文件的基本操作 373

实例 253　查看文件是否存在 373

实例 254　重命名文件 374

实例 255　复制文件夹 375

实例 256　获取文件信息 377

实例 257　获取驱动器信息 379

实例 258　读取属性文件 379

实例 259　显示指定类型的文件 381

实例 260　查找替换文本文件内容 382

实例 261　对文件夹创建、删除的操作 384

实例 262　设置 Windows 的文件属性 386

实例 263　访问类路径上的资源文件 388

实例 264　实现永久计数器 389

实例 265　从文本文件中读取注册服务条款 390

实例 266　提取文本文件内容保存到数据库 391

实例 267　将图片文件保存到数据库 393

实例 268　备份数据库文件 395

实例 269　显示数据库中的图片信息 397

实例 270　读取文件路径到数据库 399

实例 271　在数据库中建立磁盘文件索引 400

实例 272　实现文件简单的加密与解密 402

实例 273　从 XML 文件中读取数据 404

实例 274　对大文件实现分割处理 405

实例 275　将分割后的文件重新合并 407

实例 276　利用 StreamTokenizer 统计文件的字符数 408

实例 277　序列化与反序列化对象 410

11.2　无组件的文件上传 412

实例 278　单表单元素上传文件到数据库 412

实例 279　多表单元素上传文件到数据库 414

实例 280　上传文件到服务器 415

实例 281　限制文件大小的文件上传 416

11.3　通过组件实现文件上传 418

实例 282　使用 jspSmartUpload 组件实现文件上传 418

实例 283　使用 jspSmartUpload 组件实现中文名文件上传 419

实例 284　应用 jspSmartUpload 组件处理文件上传漏洞 421

实例 285　使用 commons-fileUpload 组件实现文件上传 423

实例 286　通过 commons-fileUpload 组件获取其他表单元素 424

实例 287　通过 commons-fileUpload 组件限制上传文件类型 427

11.4　文件下载 428

实例 288　利用响应输出流实现文件下载 428

实例 289　防止网站文件盗链下载 430

实例 290　隐藏文件下载的真实路径 431

实例 291　应用 jspSmartUpload 组件实现文件下载 432

实例 292　处理 jspSmartUpload 组件下载文件名乱码问题 434

第 12 章　文件的批量管理 436

12.1　文件的批量操作 437

实例 293　文件批量重命名 437

实例 294　快速批量移动文件 439

实例 295　删除指定磁盘所有.tmp 临时文件 440

实例 296　动态加载磁盘文件 442

实例 297　删除文件夹中所有文件 444

实例 298　创建磁盘索引文件 446

实例 299　快速全盘查找文件 447

实例 300　获取磁盘所有文本文件 448

实例 301　合并多个 txt 文件 450

实例 302　批量复制指定扩展名的文件 451

实例 303　将某文件夹中的文件进行分类存储 453

实例 304　在指定目录下搜索文件 454

实例 305　网络文件夹备份 456

12.2　文件的压缩与解压缩 458

实例 306　压缩所有文本文件 458

实例 307　压缩包解压到指定文件夹 459

实例 308　压缩所有子文件夹 461

实例 309　深层文件夹压缩包的释放 462

实例 310　解决压缩包中文乱码 464

实例 311　Apache 实现文件解压缩 466

实例 312　解压缩 Java 对象 467

实例 313　文件压缩为 RAR 文档 469

实例 314 解压缩 RAR 压缩包......470
实例 315 文件分卷压缩......471
实例 316 为 RAR 压缩包添加注释......473
实例 317 获取压缩包详细文件列表......474
实例 318 从 RAR 压缩包中删除文件......476
实例 319 在压缩文件中查找字符串......478
实例 320 重命名 RAR 压缩包中的文件......479
实例 321 创建自解压 RAR 压缩包......481
实例 322 设置 RAR 压缩包密码......482
实例 323 压缩远程文件夹......484
实例 324 压缩存储网页......485
12.3 文件的批量上传......487
实例 325 使用 jspSmartUpload 实现文件批量上传......487
实例 326 使用 commons-fileUpload 实现文件批量上传......488

第 3 篇 图像与多媒体篇

第 13 章 图像生成......492
13.1 绘制图形和文本......493
实例 327 绘制直线......493
实例 328 绘制矩形......494
实例 329 绘制正方形......495
实例 330 绘制椭圆......496
实例 331 绘制圆弧......497
实例 332 绘制指定角度的填充扇形......498
实例 333 绘制多边形......499
实例 334 绘制二次曲线......500
实例 335 绘制三次曲线......501
实例 336 绘制文本......503
实例 337 设置文本的字体......504
实例 338 设置文本和图形的颜色......505
13.2 绘制图案......506
实例 339 绘制五环图案......506
实例 340 绘制艺术图案......507
实例 341 绘制花瓣......509
实例 342 绘制公章......510
13.3 图形的合并运算......512
实例 343 图形的加运算......512
实例 344 图形的减运算......513
实例 345 图形的交运算......514
实例 346 图形的异或运算......515
13.4 文字特效......516
实例 347 立体效果的文字......516
实例 348 阴影效果的文字......518
实例 349 倾斜效果的文字......519
实例 350 渐变效果的文字......520
实例 351 水印文字特效......522
13.5 图片特效......523
实例 352 以椭圆形显示图像......523
实例 353 图片百叶窗特效......524
实例 354 图片马赛克特效......526
实例 355 图片的模糊效果......528
实例 356 图片的锐化效果......529
实例 357 图片的半透明效果......530
实例 358 图片的溶合效果......531
实例 359 光栅图像......532
13.6 简单的验证码应用......533
实例 360 生成中文验证码......533
实例 361 随机生成数字的验证码......536
实例 362 生成中文、英文和数字混合的验证码......538
13.7 复杂的验证码应用......540
实例 363 设置验证码的字体颜色......540
实例 364 具有背景颜色的验证码......542
实例 365 随机缩放文字并将文字旋转指定角度的验证码......543
实例 366 随机生成带有干扰线的验证码......544
实例 367 随机生成多条干扰线的验证码......546
实例 368 随机生成关键字验证码......547
实例 369 利用 Ajax 实现无刷新的彩色验证码......549
实例 370 生成带雪花的验证码......552
实例 371 生成带背景的验证码......554
13.8 生成条形码......556
实例 372 利用组件生成条形码......556
第 14 章 图像操作......559
14.1 图片的大小......560

实例 373　打开自定义大小的图片 560
14.2　图片与鼠标相关的操作 561
实例 374　当鼠标经过图片时显示图片 561
实例 375　当鼠标经过图像时给予文字提示 562
实例 376　图片的预装载 562
实例 377　按时间随机变化的网页背景 563
实例 378　左右循环滚动效果的图片 565
实例 379　浮动广告图片 566
实例 380　进度条的显示 567
实例 381　缩小与放大图片的效果 569
实例 382　通过鼠标滚轮放大与缩小图片 570
实例 383　随鼠标移动的图片 571
实例 384　左右拖动图片的效果 572
实例 385　随意拖动图片 574
实例 386　改变图片获取焦点时的状态 575
实例 387　抖动的图片 576
实例 388　鼠标移动放大图片 578
14.3　图片与时间相关的操作 580
实例 389　定时隐藏图片 580
实例 390　根据时间变换页面背景 581
实例 391　使图片不停闪烁 582
实例 392　上下跳动的图片 583
实例 393　左右晃动的图片 585
实例 394　移动变形的图片 586
14.4　图片的动画效果 589
实例 395　图片翻转效果 589
实例 396　图片的水波倒影效果 590
实例 397　图片渐隐渐现 591
实例 398　图片的探照灯效果 592
实例 399　雷达扫描式图片效果 593
实例 400　在页面中旋转的图片效果 594
实例 401　改变形状的图片 595
14.5　选择头像图片 596
实例 402　在列表中选择图片头像 597
实例 403　在弹出的新窗口中选择图片 598
14.6　图片的其他效果 599
实例 404　页面中播放图片 599
实例 405　导航地图 601
第 15 章　多媒体应用 603
15.1　播放音乐 604
实例 406　为网页设置背景音乐 604
实例 407　随机播放背景音乐 605
实例 408　MIDI 音乐选择 606
实例 409　在线连续播放音乐 607
实例 410　同步显示 LRC 歌词 611
实例 411　把显示后的 LRC 歌词变换颜色 615
15.2　插入 Flash 动画 616
实例 412　插入 Flash 动画 616
实例 413　插入背景透明的 Flash 动画 617
15.3　播放视频 618
实例 414　播放视频文件 618
实例 415　自制视频播放器 620
实例 416　在线播放 FLV 视频 621

第 4 篇　窗体应用篇

第 16 章　窗口的应用 624
16.1　弹出窗口控制 625
实例 417　打开网页显示广告信息 625
实例 418　定时关闭广告窗口 626
实例 419　弹出窗口的居中显示 627
实例 420　通过按钮创建窗口 628
实例 421　为弹出的窗口加入关闭按钮 629
实例 422　定时打开窗口 630
实例 423　关闭弹出窗口时刷新父窗口 631
实例 424　关闭窗口时不弹出询问对话框 632
实例 425　弹出窗口的 Cookie 控制 633
16.2　弹出网页对话框 634
实例 426　弹出网页模式对话框 634
实例 427　全屏显示网页模式对话框 635
实例 428　实现网页日期选择 636
实例 429　网页拾色器 641
16.3　窗口的动画效果 643
实例 430　页面自动滚动 643
实例 431　动态显示网页 644
实例 432　指定窗口的扩展大小 645
实例 433　实现空降窗口 646
实例 434　慢慢变大窗口 647

实例 435　移动的窗口 648
实例 436　震颤窗口 649
实例 437　旋转的窗口 651
16.4　窗口控制 652
实例 438　始终将窗口居上显示 652
实例 439　窗口全屏显示 653
实例 440　自动最大化窗口 654
实例 441　按钮实现最大和最小化 655
实例 442　频道方式的窗口 656
实例 443　根据用户分辨率自动调整窗口 657
实例 444　使窗口背景透明 658
16.5　框架的应用 659
实例 445　框架集的嵌套 659
实例 446　在网页中应用浮动框架 661
实例 447　创建空白框架 663
实例 448　居中显示框架 665
16.6　无边框窗口 666
实例 449　全屏显示无边框有滚动条的窗口 666
实例 450　应用 CSS 实现指定尺寸无边框无滚动条窗口 667
实例 451　应用 JavaScript 实现指定尺寸无边框无滚动条窗口 669

第 17 章　导航条的应用 671
17.1　水平导航条的应用 672
实例 452　带图标的文字导航条 672
实例 453　Flash 导航条 673
实例 454　图片按钮导航条 674
实例 455　导航条的动画效果 675
实例 456　动态改变导航菜单的背景颜色 676
实例 457　不用图片实现质感导航条 677
实例 458　标签页导航条 678
17.2　下拉菜单式导航条 680
实例 459　二级导航菜单 680
实例 460　半透明背景的下拉菜单 681
实例 461　弹出式下拉菜单 684
实例 462　弹出式悬浮菜单 686
实例 463　应用 setTimeout()函数实现展开式导航条 687
实例 464　应用 setInterval()函数实现展开式导航条 688
实例 465　用层制作下拉菜单 1 689
实例 466　用层制作下拉菜单 2 690
17.3　侧导航条设计 692
实例 467　收缩式导航菜单 692
实例 468　树状导航菜单 694
实例 469　自动隐藏的弹出式菜单 696

第 18 章　表单的应用 698
18.1　文本框/编辑框/隐藏域组件 699
实例 470　获取文本框/编辑框/隐藏域的值 699
实例 471　自动预算 700
实例 472　设置文本框为只读属性 702
实例 473　限制文本域字符个数 703
实例 474　自动选择文本框和编辑框的文字 704
实例 475　按 Enter 键时自动切换焦点 705
18.2　下拉列表与菜单的应用 706
实例 476　获取下拉列表、菜单的值 706
实例 477　遍历多选下拉列表 707
实例 478　在下拉列表中进行多选择移除 708
实例 479　将数组中的数据添加到下拉菜单中 709
实例 480　下拉菜单选择所要联机的网站 710
实例 481　多级级联菜单 711
实例 482　分级下拉列表 712
18.3　单选按钮 713
实例 483　不提交表单获取单选按钮的值 713
实例 484　选中单选按钮后显示其他表单元素 714
实例 485　通过单选按钮控制其他表单元素是否可用 715
18.4　复选框 716
实例 486　只有一个复选框时控制复选框的全选或反选 716
18.5　密码域 717
实例 487　让密码域更安全 717
实例 488　不提交表单自动检测密码域是否相同 718
18.6　表单的应用 718
实例 489　通过 JavaScript 控制表单的提交与重置 719
实例 490　带记忆功能的表单 719
实例 491　防止表单重复提交 720
实例 492　自动提交表单 721
实例 493　通过 for 循环获取表单元素的中文名称 722
实例 494　可以提交到不同处理页的表单 723

第 19 章　表格的操作 724
19.1　应用 JavaScript 操作表格 725
实例 495　动态制作表格 725
实例 496　删除表中的行 727

实例 497　动态生成行或列......728
实例 498　合并单元格......730
实例 499　在表格中添加行及单元格......731
实例 500　删除表中的单元格......732
实例 501　从表格最下面向上删除单元格......733
实例 502　在表格的右侧动态添加列......734
实例 503　从表格的右侧依次删除所有列......735
实例 504　在表格中动态添加行......736
19.2　对单元格进行控制......737
实例 505　选定表格中的单元格......737
实例 506　可左右移动单元格的信息......738
实例 507　使用键盘使单元格焦点随意移动......740
实例 508　隐藏及显示单元格......744
实例 509　编辑单元格中的文本信息......745
实例 510　单元格外边框加粗......747
19.3　表格的特殊效果......748
实例 511　闪烁的表格边框......748
实例 512　选中行的变色......749
实例 513　表格中表元内部空白......749
实例 514　表格中表元间隙......750
实例 515　对表格内文字进行对齐......751
实例 516　对表格内信息进行布局......751
实例 517　对表格的大小进行设置......752
实例 518　透明表格......753
实例 519　限制表格的宽度......754
实例 520　表格的标题......755
实例 521　表格的外阴影......755
实例 522　立体表格......756
实例 523　虚线边框表格......757
实例 524　表格作为分割线......758
实例 525　表格向下展开......759
实例 526　表格向右拉伸......760

第 5 篇　操作 Word、Excel、报表与打印篇

第 20 章　JSP 操作 Word......764
20.1　应用 JavaScript 导出到 Word......765
实例 527　将 JSP 页面的信息在 Word 中打开......765
20.2　应用响应流导出到 Word......766
实例 528　将表单数据输出到 Word 中......766
实例 529　将查询结果输出到 Word 中......768
实例 530　将页面中的学生表以 Word 表格保存......770
20.3　应用 POI 组件导出到 Word......772
实例 531　将数据库中的数据写入到 Word 中......772
第 21 章　JSP 操作 Excel......775
21.1　应用 JXL 组件操作 Excel......776
实例 532　创建 Excel 工作表......776
实例 533　将表单信息导出到 Excel......777
实例 534　向 Excel 工作表中添加数值......779
实例 535　向 Excel 工作表中添加格式化数值......781
实例 536　向 Excel 工作表中添加 boolean 值......782
实例 537　向 Excel 工作表中添加日期时间......783
实例 538　向 Excel 工作表中添加格式化日期时间......784
实例 539　设置 Excel 工作表字体样式......786
实例 540　合并 Excel 工作表的单元格......787
实例 541　设置 Excel 工作表的单元格内容水平居中......788
实例 542　设置 Excel 工作表的行高......790
实例 543　设置 Excel 工作表的列宽......791
实例 544　设置 Excel 工作表的单元格内容垂直居中......792
实例 545　设置 Excel 工作表的单元格内容自动换行......794
实例 546　设置 Excel 工作表的单元格样式......795
实例 547　向 Excel 工作表中插入图片......797
实例 548　将数据库数据导出到 Excel......798
实例 549　读取 Excel 中的数据和图片并保存到数据库......800
实例 550　设置 Excel 工作表简单的打印属性......803
实例 551　设置 Excel 工作表详细的打印属性......805
21.2　应用 POI 组件操作 Excel......807
实例 552　创建 Excel 文档......807
实例 553　在 Excel 工作表中创建单元格......808
实例 554　向 Excel 单元格中添加不同类型的数据......810
实例 555　创建指定格式的单元格......811
实例 556　设置单元格内容的水平对齐方式......813
实例 557　设置单元格内容的垂直对齐方式......814
实例 558　合并单元格......816
实例 559　设置单元格的边框样式......817
实例 560　设置字体样式......819
实例 561　向 Excel 文件中插入图片......820

实例 562　将数据库数据导出到 Excel 文件 822
实例 563　读取 Excel 文件的数据到数据库 824
实例 564　设置 Excel 文件的打印属性 826

第 22 章　报表与打印 829
22.1　Web 打印 830
实例 565　利用 JavaScript 调用 IE 自身的打印功能 830
实例 566　利用 WebBrowser 打印 831
实例 567　打印分组报表 831
22.2　利用 Word 打印报表 833
实例 568　将页面中的客户列表导出到 Word 并打印 833
实例 569　利用 Word 自动打印指定格式的会议记录 835
实例 570　利用 Word 生成的 HTML 实现打印 836
22.3　利用 Excel 打印报表 838
实例 571　利用 Excel 打印工作报表 838
实例 572　将页面数据导出到 Excel 并自动打印 840
22.4　应用 WebBrowser+CSS 套打邮寄产品单 841
实例 573　打印汇款单 841
实例 574　打印信封 843
22.5　打印库存报表 844
实例 575　打印库存明细表 844
实例 576　打印库存盘点报表 846
实例 577　打印库存汇总报表 847
实例 578　打印指定条件的库存报表 849
22.6　高级报表 850
实例 579　应用 iReport+JasperReport 生成主从报表 850
实例 580　应用 iReport+JasperReport 生成分栏报表 856

第 6 篇　综合应用篇

第 23 章　综合应用 862
23.1　在线投票系统 863
实例 581　禁止重复投票的在线投票系统 863
实例 582　每个 IP 一个月只能投票一次的投票系统 864
23.2　用户注册 865
实例 583　带检测用户名的用户注册 865
实例 584　分步用户注册 867
实例 585　通过 E-mail 激活的用户注册 869
23.3　论坛 873
实例 586　查看帖子信息 873
实例 587　发表主题信息 875
实例 588　回复主题信息 876
实例 589　删除主题及回复信息 877
实例 590　注销用户 878
23.4　购物车 879
实例 591　添加至购物车 879
实例 592　查看购物车 881
实例 593　修改商品购买数量及从购物车中移除指定商品 882
实例 594　清空购物车 883
实例 595　收银台结账 884
23.5　聊天室 885
实例 596　Application 形式的聊天室 885
实例 597　带私聊的聊天室 888
实例 598　XML 形式的聊天室 895
23.6　万年历 901
实例 599　简易万年历 901
实例 600　带阴历的万年历 903

第 1 篇

基础篇

⏭ 第 1 章　开发环境搭建

⏭ 第 2 章　Java 语言基础

⏭ 第 3 章　HTML/CSS 技术

⏭ 第 4 章　JSP 基础与内置对象

⏭ 第 5 章　JavaBean 技术

⏭ 第 6 章　Servlet 技术

⏭ 第 7 章　过滤器与监听器技术

⏭ 第 8 章　JSTL 标签库

⏭ 第 9 章　JavaScript 技术

⏭ 第 10 章　Ajax 技术

第 1 章

开发环境搭建

- JDK 开发工具包
- Tomcat 服务器
- Linux 系统配置 JDK 与 Tomcat 服务器

1.1　JDK 开发工具包

“工欲善其事，必先利其器。”在学习一门编程语言之前，通常情况下都需要安装编程语言的开发环境，学习 Java Web 也不例外，需要 Java 配置的运行环境与安装测试 JDK 开发包。

JDK 是 Java Development Kit 的缩写，由于 Sun 公司已经被 Oracle（甲骨文）公司所收购，所以现在可以说是 Oracle 公司提供的 JDK，其中包括运行 Java 程序所必需的 JRE 环境及开发过程中常用的 Java 基本类库文件。在开发 Java Web 应用之前，首先应该安装 JDK 组件。

实例 001　JDK 的下载

初级　实用指数：★

实例说明

开发 Java 程序必须有 Java 开发环境，即 JDK 开发工具包，这个工具包包含了编译、运行、调试等关键的命令。哪怕运行 Eclipse、NetBeans 等开发工具也要有 JDK 或 JRE 的支持，所以开发 Java 程序之前的第一步准备工作就是获取 JDK 开发工具包，该工具包需要到官方网站去下载，本实例将介绍其关键的下载步骤。首先要打开浏览器并浏览 JDK 的下载页面，如图 1.1 所示。

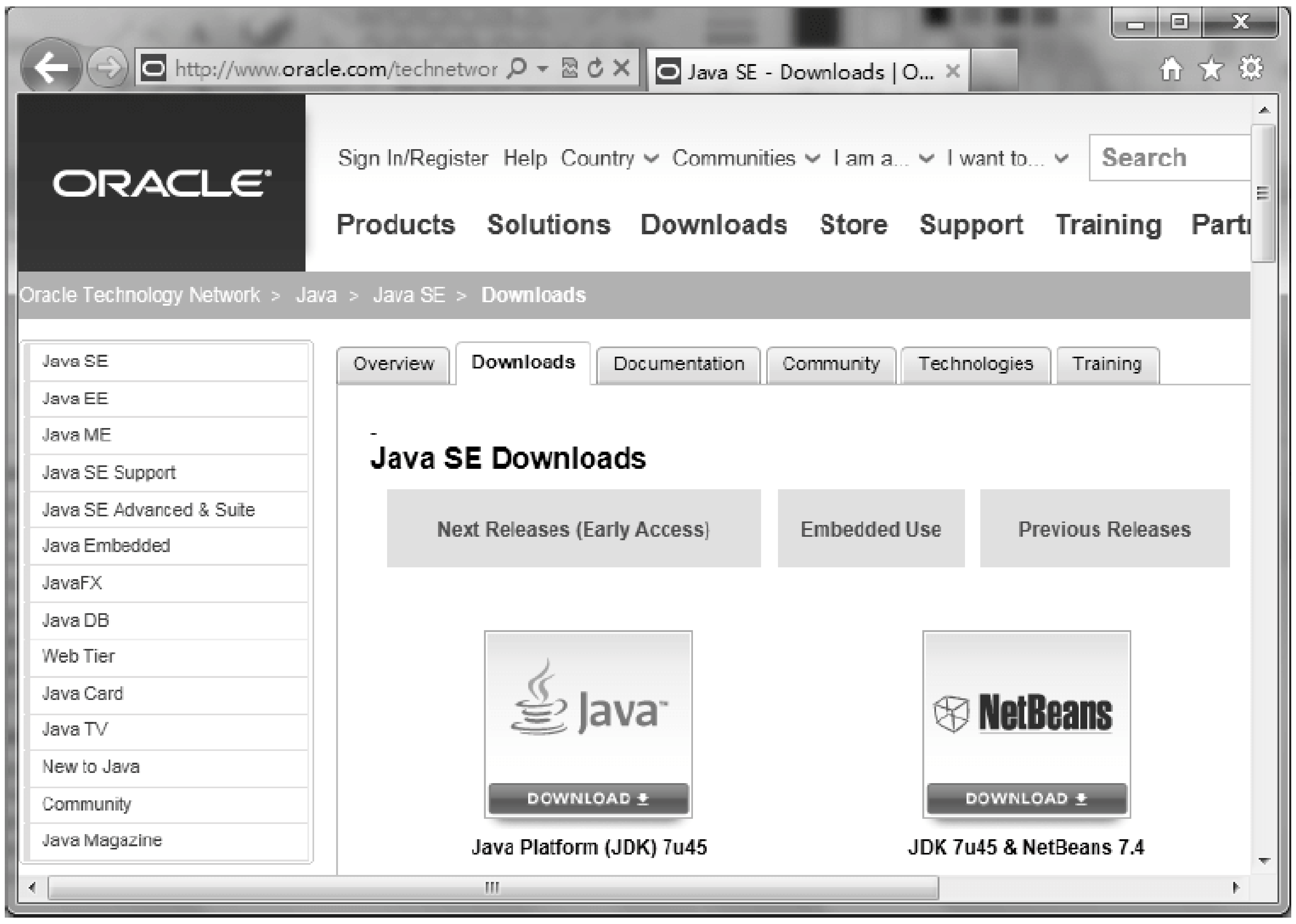


图 1.1　JDK 下载页面

关键技术

现在的 Java 属于 Oracle 公司，而且在下载页面中也会有 Oracle 公司的标志，这里要做的就是找到 Java SE 的下载网址。而在 Oracle 官方网站的主页上可以看到一个 Downloads 选项卡，通过这个选项卡可以找到相应的下载页面。

设计过程

由于推出 JDK 的 Sun 公司已经被 Oracle 公司收购了，所以 JDK 可以到 Oracle 官方网站（http://www.oracle.com/index.html）中下载。下面以目前最新版本的 JDK 7 Update 45 为例介绍下载 JDK 的方法，具体下载步骤如下。

（1）打开 IE 浏览器，在地址栏中输入 URL 地址“http://www.oracle.com/index.html”，并按下 Enter 键，进入到 Oracle 官方网站的主页。在该页面中，将鼠标移动到 Downloads 选项卡上，将显示如图 1.2 所示的内容。



图 1.2　Oracle 官方主页

（2）在 Downloads 选项卡的 Popular Downloads 栏中，单击 Java for Developers 超链接，进入到如图 1.3 所示的 Java SE 下载页面。



图 1.3　Java SE 下载页面

说明：在 JDK 中，已经包含了 JRE。JDK 用于开发 Java 程序，JRE 用于运行 Java 程序。

（3）在图 1.3 所示的页面中，单击 Java Platform（JDK）7u45 上方的 DOWNLOAD 按钮，将进入到如图 1.4 所示的下载列表页面，选中 Accept License Agreement 单选按钮同意协议后，将显示如图 1.5 所示的页面，这时可以单击当前系统对应的 JDK 下载超链接，下载适合当前系统的 JDK。例如，要安装在 32 位的 Windows 操作系统中，可以下载 jdk-7u45-windows-i586.exe 文件。

秘笈心法

心法领悟 001：应对可能变化的下载页面。

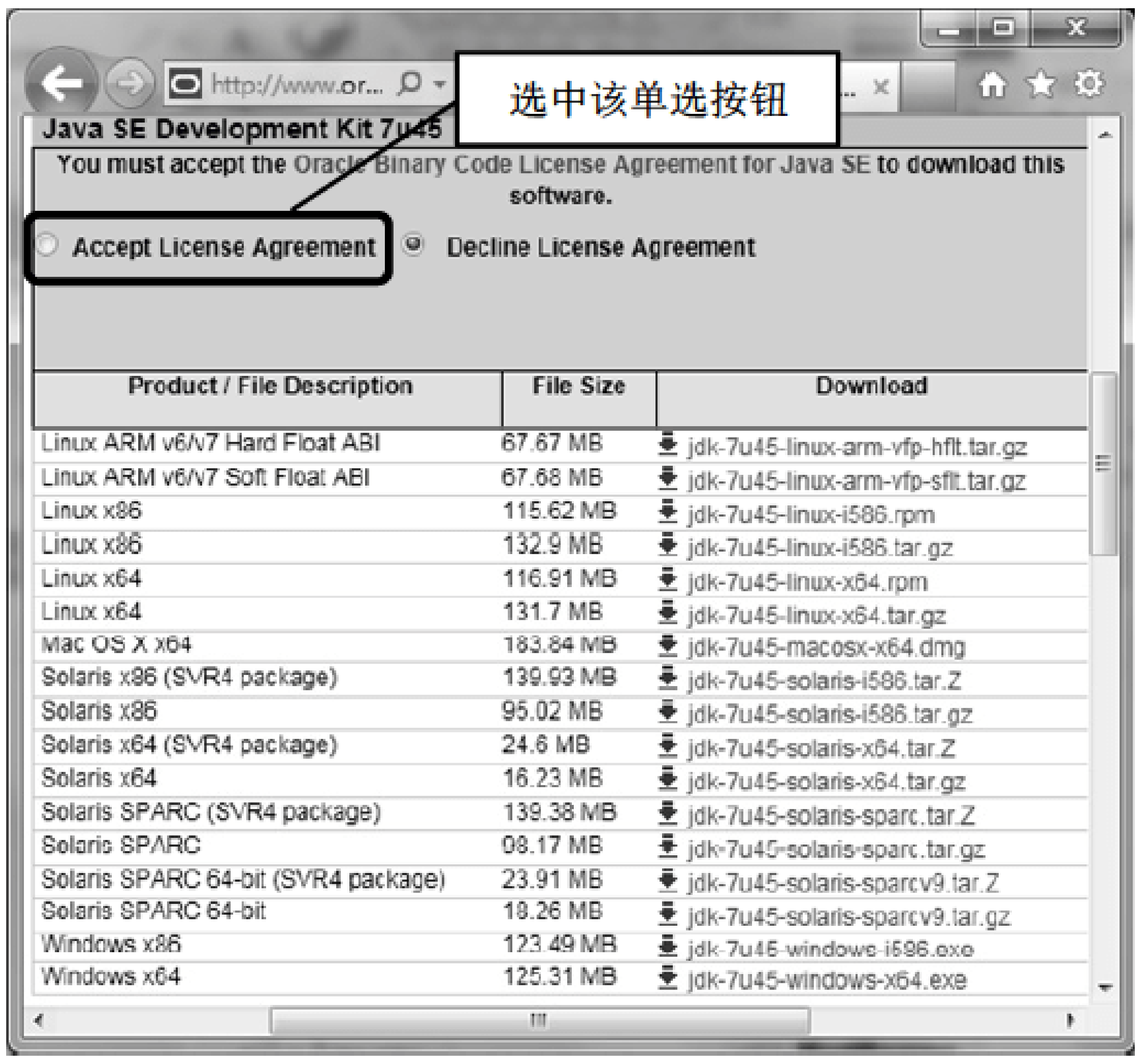


图 1.4 JDK 资源选择页面

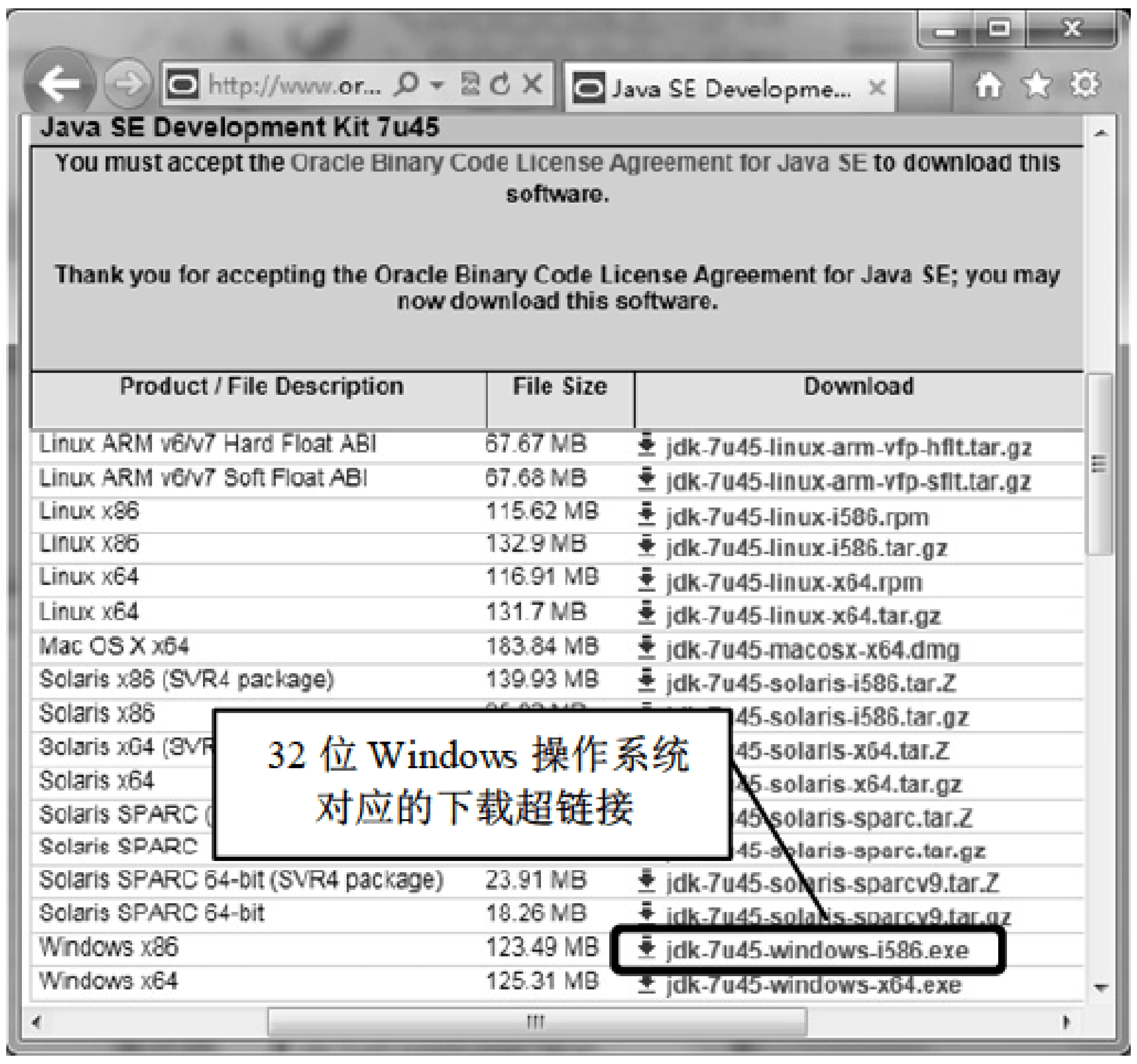


图 1.5 接受许可协议后的页面

在下载 Java SE 的 JDK 安装文件时，由于时间的推移和网页不断地改进，下载页面可能会发生一些变化，但是无论如何改变网页的布局，只要记住在网页中找到 Java SE 的资源网页，然后在其中找到 Downloads 超链接，通过这个超链接找到 JDK 的下载页面，并在页面中进行简单设置，再选择要下载哪个平台的 JDK 安装文件，并执行下载任务即可。

实例 002 JDK 的安装

初级

实用指数：★

实例说明

安装 JDK 开发工具包意味着编写 Java 程序的开始。在一台计算机中安装 JDK，可以为计算机增加编译、运行和调试 Java 程序的能力。本实例将介绍如何安装 JDK 开发工具包到指定的磁盘位置，这比简单的默认安装要稍微复杂一些，但是这样能够详细地了解安装的步骤。JDK 安装向导启动界面如图 1.6 所示。



图 1.6 JDK 安装向导启动界面

关键技术

在安装 JDK 开发工具包时应注意，系统中已经安装的某些杀毒软件或者系统防范工具对安装的提示信息，因为 JDK 开发工具包会在系统中添加一些方便以后升级的启动项，在杀毒软件提示是否允许这项操作时，请让它通过，或者干脆暂时关闭杀毒软件，以确保 JDK 能够完整地安装，并随时保持可升级状态。

设计过程

在网站下载的 JDK 安装向导根据版本的不同，安装文件的名称也有所改变。这里以 jdk-7u45-windows-i586.exe 安装文件为例介绍安装过程，首先运行这个安装文件。

（1）双击 JDK 的安装文件，将弹出如图 1.7 所示的欢迎对话框。

（2）单击“下一步”按钮，将弹出“自定义安装”对话框，在该对话框中，可以选择安装的功能组件，这

里选择默认设置，如图 1.8 所示。



图 1.7　欢迎对话框

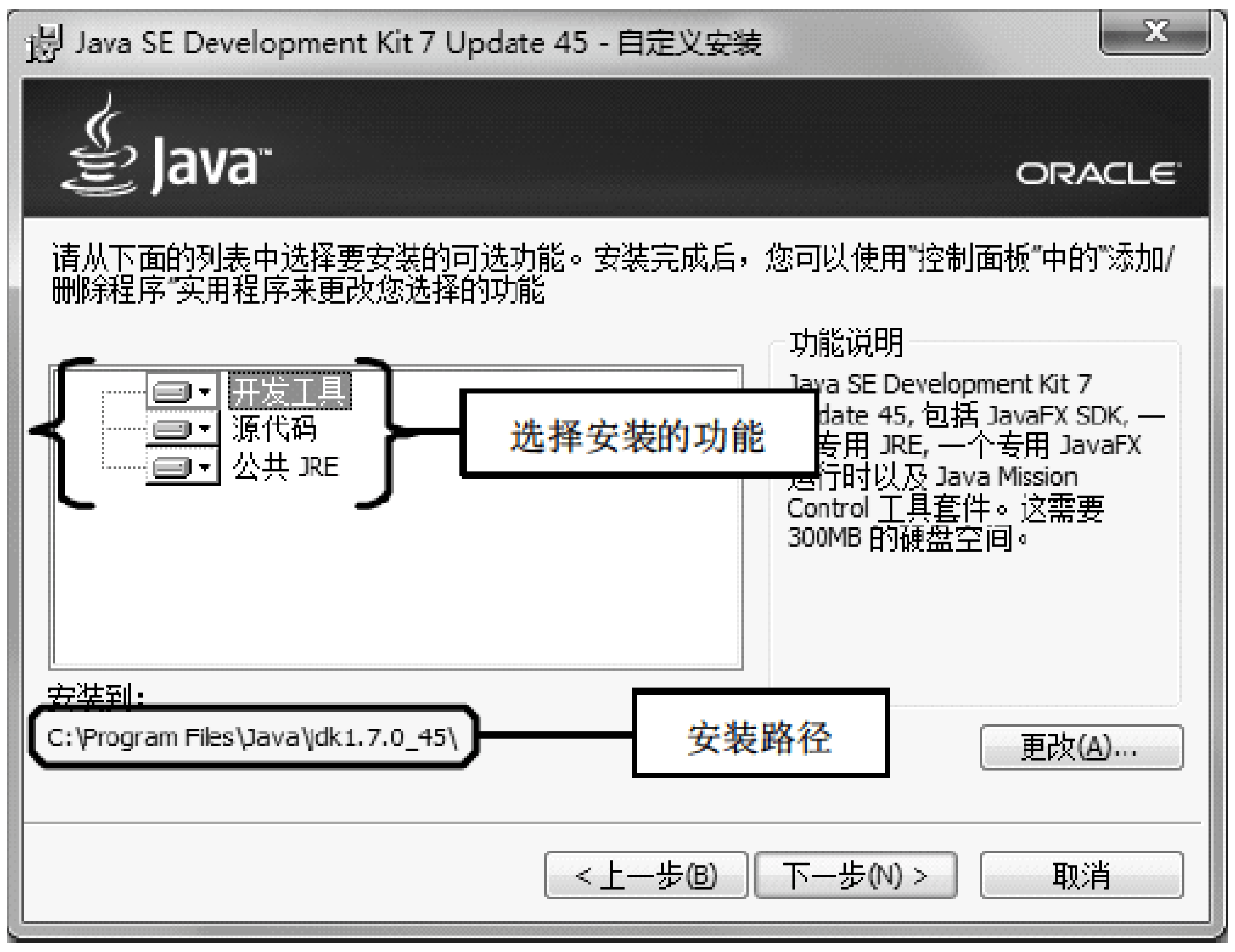


图 1.8　JDK“自定义安装”对话框

（3）单击“更改”按钮，将弹出更改文件夹的对话框，在该对话框中将 JDK 的安装路径更改为 C:\Java\jdk1.7.0_45\，如图 1.9 所示，单击“确定”按钮，将返回到“自定义安装”对话框中。

（4）单击“下一步”按钮，开始安装 JDK。在安装过程中会弹出 JRE 的“目标文件夹”对话框，这里更改 JRE 的安装路径为 C:\Java\jre7\，然后单击“下一步”按钮，安装向导会继续完成安装进程。

说明：JRE 全称为 Java Runtime Environment，它是 Java 运行环境，主要负责 Java 程序的运行，而 JDK 包含了 Java 程序开发所需要的编译、调试等工具，另外还包含了 JDK 的源代码。

（5）安装完成后，将弹出如图 1.10 所示的对话框，单击“后续步骤”按钮，将联网访问教程、API 文档、开发人员指南等内容，如果不想查看，可以单击“关闭”按钮，完成 JDK 的安装。



图 1.9　更改 JDK 的安装路径对话框



图 1.10　完成对话框

秘笈心法

心法领悟 002：JDK 与 JRE 的区别。

在 JDK 开发工具包的安装向导中包含了 JRE，而 JRE 到底是什么？它和 JDK 有什么区别？

这个问题可以从名字上进行区分，JDK 的意义是“Java 开发工具”，而 JRE 的意义是“Java 运行时环境”，也就是说，JDK 负责开发程序，因为它拥有代码编译、调试和运行的所有命令。JRE 是负责运行 Java 程序的，当然是经过编译后的 Java 程序。JRE 只能运行 Java 程序的命令与一些类库等其他资源，所以它的体积要比 JDK 小很多。而 JDK 中集成 JRE 是为了在系统中提供 Java 运行环境，虽然 JDK 也有运行 Java 的命令，但是它不像 JRE 那样与操作系统集成，并可以直接使用命令，JDK 需要经过环境变量的设置才能像 JRE 那样。

实例 003　设置 Java 环境变量

光盘位置：光盘\MR\01\003

初级

实用指数：★

实例说明

在 JDK 安装完成之后，最为重要的步骤就是设置它运行时的 Windows 环境变量。只有设置了环境变量之后，在应用程序中才能正常使用 JDK 中提供的类库和 JRE 运行环境。

关键技术

在为 JDK 配置环境变量的过程中，需要注意的是，设置 path 属性的变量值，一定不要替换原来 path 变量的值，因为里面配置的值可能是系统其他应用程序的 path 值，替换掉可能会导致其他程序不能正常运行。

设计过程

（1）在“开始”菜单的“计算机”图标上单击鼠标右键，在弹出的快捷菜单中选择“属性”命令，在弹出的“属性”对话框左侧单击“高级系统设置”超链接，将出现如图 1.11 所示的“系统属性”对话框。

（2）单击“环境变量”按钮，将弹出“环境变量”对话框，如图 1.12 所示，单击“系统变量”栏中的“新建”按钮，创建新的系统变量。

注意： 新建环境变量时，一定要确认是在“系统变量”列表框中新建，这样新建的环境变量在整个系统中都起作用。

（3）弹出“新建系统变量”对话框，分别输入变量名“JAVA_HOME”和变量值（即 JDK 的安装路径），其中变量值是笔者的 JDK 安装路径，读者需要根据自己的计算机环境进行修改，如图 1.13 所示。单击“确定”按钮，关闭“新建系统变量”对话框。

（4）在系统变量中查看 CLASSPATH 变量，如果不存在，则新建变量 CLASSPATH，变量的值为：

```
.;%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar
```

注意： JAVA_HOME 变量的变量值一定要确保其路径的正确性，所以建议从 JDK 实际安装路径去复制并粘贴到“变量值”文本框中，以确保其变量值与 JDK 的对应。CLASSPATH 变量中的%JAVA_HOME%是对 JAVA_HOME 变量值的引用形式。

（5）在图 1.12 所示的“环境变量”对话框中双击 Path 变量对其进行修改，在原变量值最前端添加“.;%JAVA_HOME%\bin;”变量值（注意：最后的“;”不要丢掉，它用于分割不同的变量值），如图 1.14 所示。单击“确定”按钮完成环境变量的设置。



图 1.11　“系统属性”对话框



图 1.12　“环境变量”对话框



图 1.13　“新建系统变量”对话框



图 1.14　设置 Path 环境变量值

注意：不能删除系统变量 Path 中的原有变量值，并且“%JAVA_ HOME%\bin”与原有变量值之间用英文半角的“;”号分隔，否则会产生错误。

实例 004 使用命令行工具测试 JDK

初级

实用指数：★

实例说明

在安装和配置 JDK 之后，本实例将应用 Windows 系统的命令行工具 cmd.exe 来测试 JDK 是否安装成功。

设计过程

（1）选择“开始”/“运行”命令，在打开的对话框中输入“cmd”命令，将进入 DOS 环境，在命令提示符中直接输入“javac”命令，查看是否有信息提示，如图 1.15 所示。

（2）测试 Oracle Java Mission Control，它是一个用于对 Java 应用程序进行管理、监视、概要分析和故障排除的工具套件。选择“开始”/“运行”命令，在弹出的对话框中输入“jmc”后按 Enter 键，打开如图 1.16 所示的窗口。

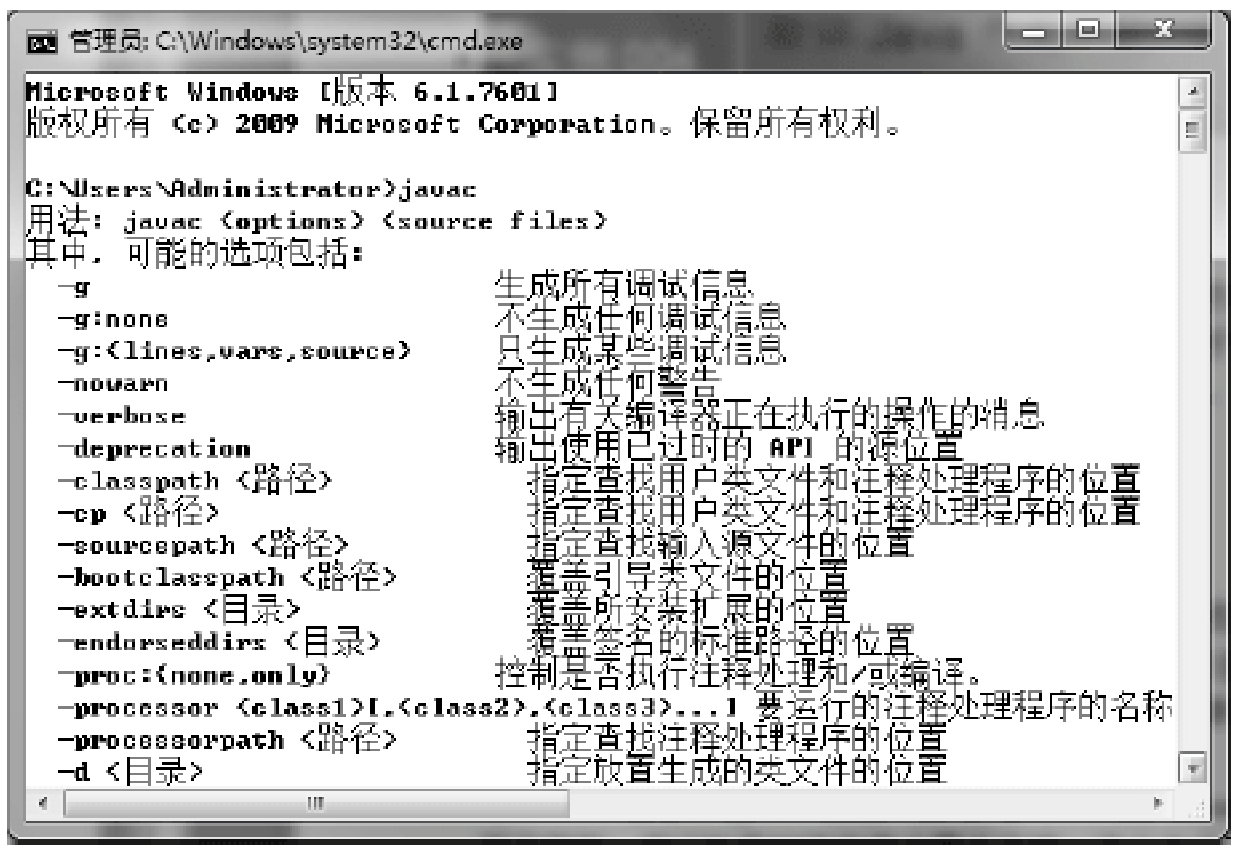


图 1.15　测试 JDK 配置是否成功



图 1.16　Oracle Java Mission Control 工具套件

实例 005 在命令行编译 Java 源码

光盘位置：光盘\MR\01\005

初级

实用指数：★

实例说明

用记事本编写一个简单的*.java 源码文件，并运行命令提示符工具，在其中输入编译 Java 文件的命令，并输出“Hello Word!”。

关键技术

在调试 JDK 时，可以在命令行提示符工具中输入“javac –version”命令来查看 JDK 的相关信息与命令，javac 命令用于将*.java 源文件编译转换成*.class 文件，然后再用 Java 命令运行编译后的文件。

设计过程

（1）选择“开始”/“运行”命令，在打开的对话框中输入“cmd”，将弹出命令行工具的控制台界面。

（2）编译文件。例如，在 C 盘根目录下有一个名为 HelloWord.java 的 Java 文件，将它编译为 class 文件。在控制台中输入“javac HelloWord.java”，并按 Enter 键之后，Java 源文件将被编译为 class 文件。可以发现，会在 C 盘根目录下自动生成一个名为 HelloWord.class 的文件，这就是编译后的文件，如图 1.17 所示。

（3）在控制台中继续输入命令“java HelloWord”后，会输出“HelloWord!”字符串，如图 1.18 所示。

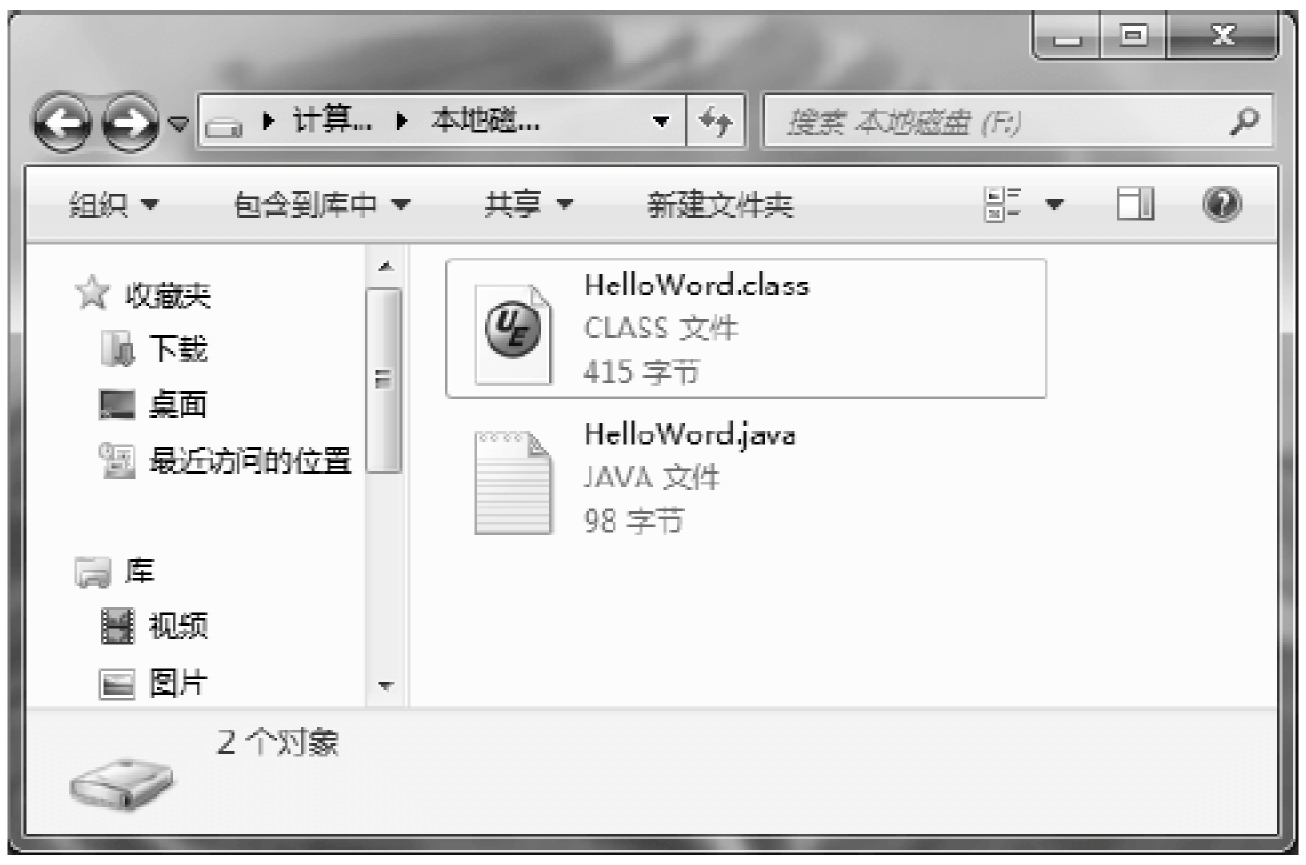


图 1.17　编译后自动生成的.class 文件



图 1.18　控制台执行编译后的文件结果

1.2　Tomcat 服务器

Tomcat 服务器是 Apache Jakarta 项目组开发的产品。当前的最新版本是 Tomcat 8，它能够支持 Servlet 3.1 和 JSP 2.3 规范，并且具有免费和跨平台等诸多特性，是学习开发 Java Web 应用的首选服务器。由于到目前为止，Tomcat 8 还是 Alpah 版，不推荐在产品中使用，所以本书中采用的是 Tomcat 7.0 版本，该版本是一个比较成熟的版本。下面将以 Tomcat 7.0.47 为例介绍下载与调试 Tomcat 的具体步骤。

实例 006　下载 Tomcat 服务器

初级

实用指数：★

实例说明

Tomcat 归于 Apache 基金组织下，其应用范围很广泛，既可用来学习 Java Web 开发，也可以用来构架各类中小型商业应用。本书使用的 Tomcat 7 已经实现了 Java EE 6 中 Web 层的各种规范。

设计过程

（1）在 IE 浏览器地址栏中输入“http://tomcat.apache.org”，进入到 Tomcat 官方网站，如图 1.19 所示。

（2）在左侧的 Download 列表中可以下载 Tomcat 服务器的各种版本。单击 Tomcat 7.0 超链接，进入到 Tomcat 7.0 版本的下载页面中，找到如图 1.20 所示的下载位置。



图 1.19　Tomcat 官方网站首页

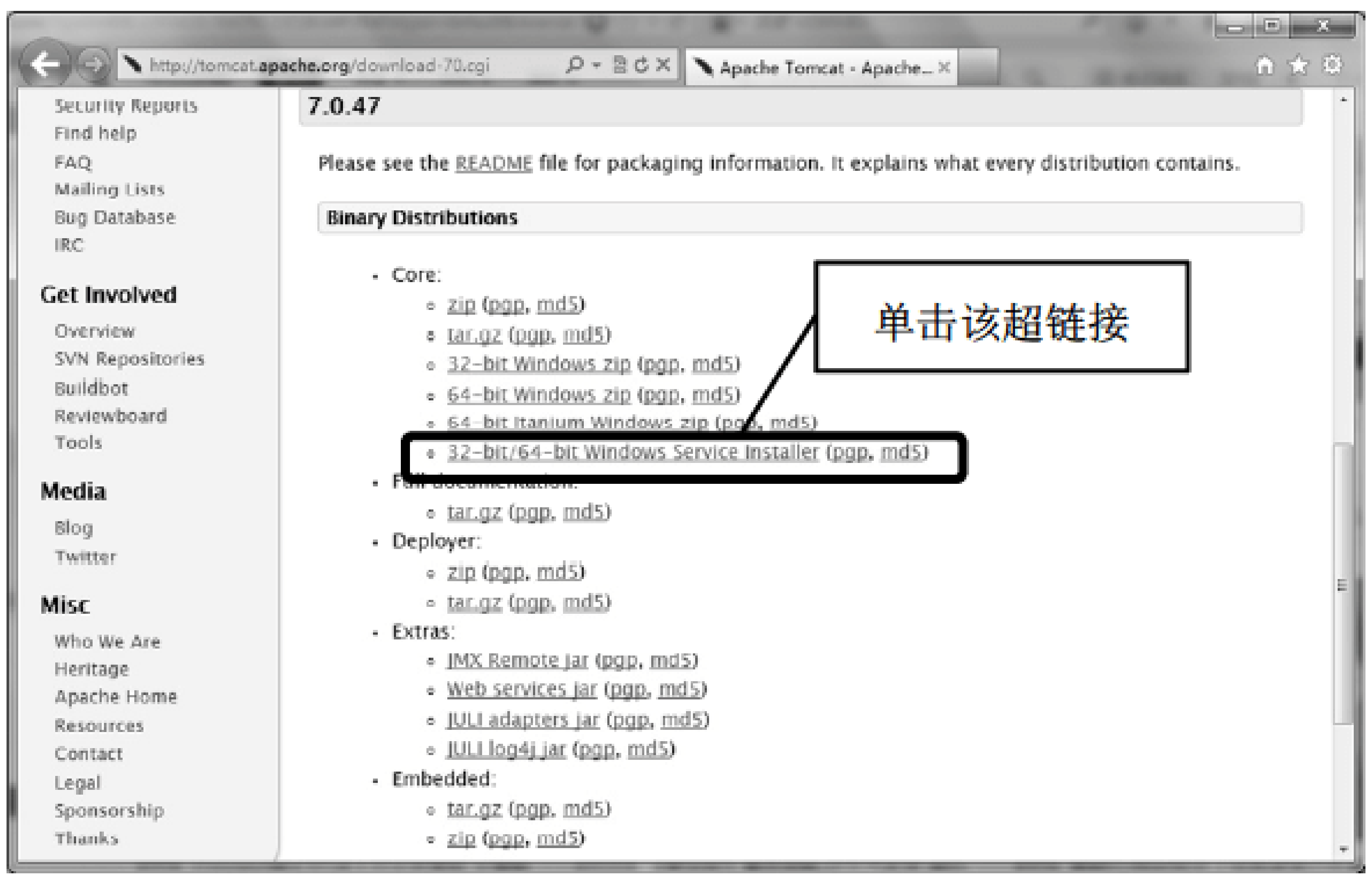


图 1.20　Tomcat 7.0 的下载页面

（3）在图 1.17 中，在 Core 节点中包含了 Tomcat 7.0 服务器安装文件的不同平台下的不同版本，此处单击的是 32-bit/64-bit Windows Service Installer(pgp,md5)超链接，单击该链接可下载安装版本的 Tomcat，然后将打开文件下载对话框，在对话框中单击“保存”按钮，即可将 Tomcat 的安装文件下载到本地计算机中。

实例 007 安装 Tomcat 服务器

初级

实用指数：★

实例说明

本实例将介绍如何安装 Tomcat 服务器。Tomcat 安装文件下载完毕后，就可以通过该文件安装 Tomcat 服务器。

设计过程

（1）双击 apache-tomcat-7.0.47.exe，弹出安装向导界面（如图 1.21 所示），单击 Next 按钮，打开 Apache 的许可协议对话框。

（2）单击 I Agree 同意协议，打开 Apache 的主程序与组件安装对话框，在该对话框中选择要安装的组件，如图 1.22 所示。

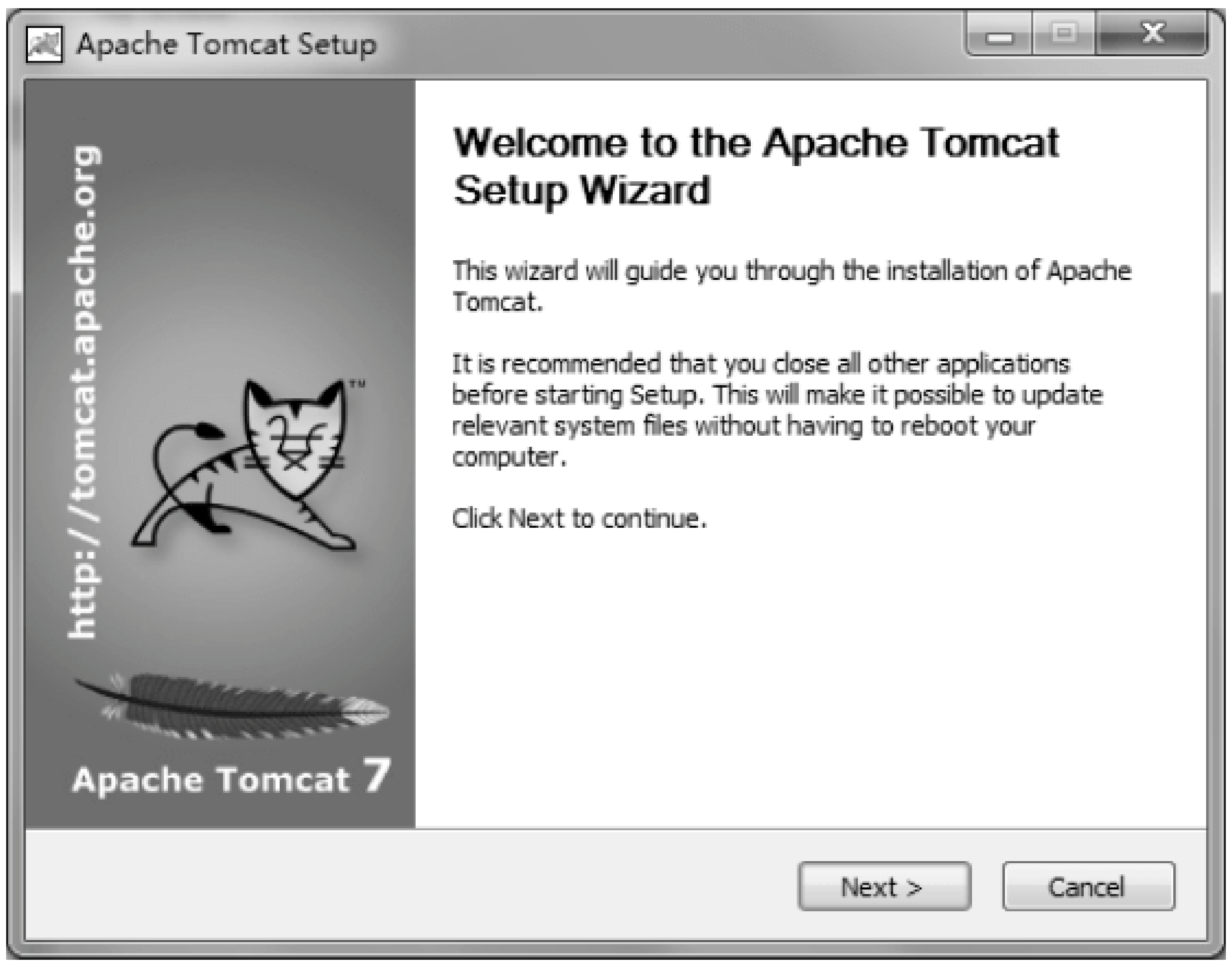


图 1.21　安装向导界面

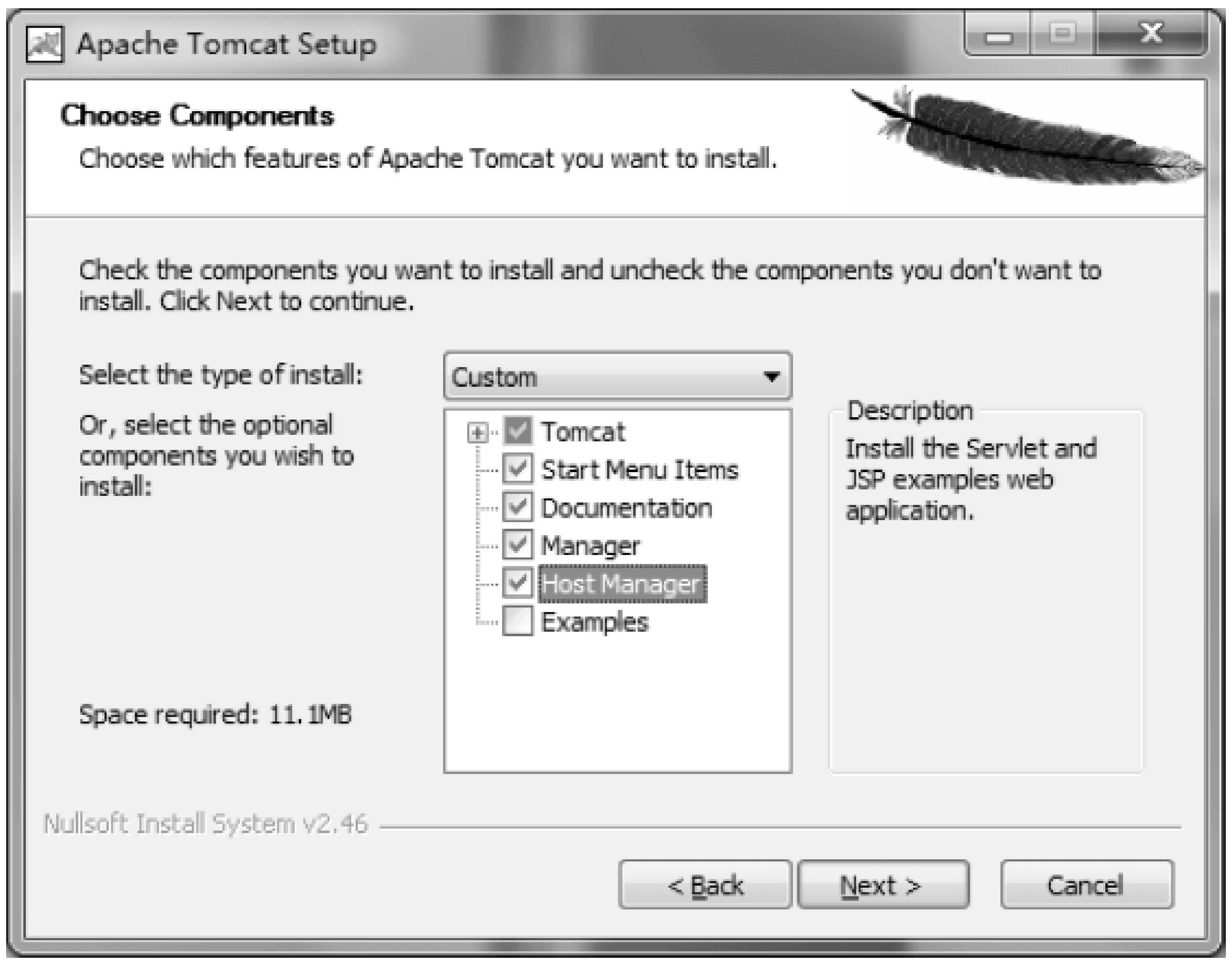


图 1.22　Apache 的主程序与组件安装对话框

（3）单击 Next 按钮，在打开的对话框中设置访问 Tomcat 服务器的端口及用户名和密码，通常设置端口为 8080、用户名为 admin、密码为 111，如图 1.23 所示。

说明： 一般情况下，不要修改默认的端口号，除非 8080 端口已经被占用。

（4）单击 Next 按钮，在打开的 Java Virtual Machine 对话框中选择 Java 虚拟机路径，这里选择 JDK 的安装路径，如图 1.24 所示。

（5）单击 Next 按钮，将打开 Choose Install Location 对话框。在该对话框中可通过单击 Browse 按钮更改 Tomcat 的安装路径，这里将其更改为 C:\Program Files\Tomcat 7.0 目录下，如图 1.25 所示。

（6）单击 Install 按钮，开始安装 Tomcat。在打开安装完成的提示对话框中，取消选中 Run Apache Tomcat 和 Show Readme 复选框，单击 Finish 按钮，即可完成 Tomcat 的安装。

（7）Tomcat 服务器安装成功后，在 Tomcat 的安装目录下会出现 7 个文件夹和 4 个文件，如图 1.26 所示。

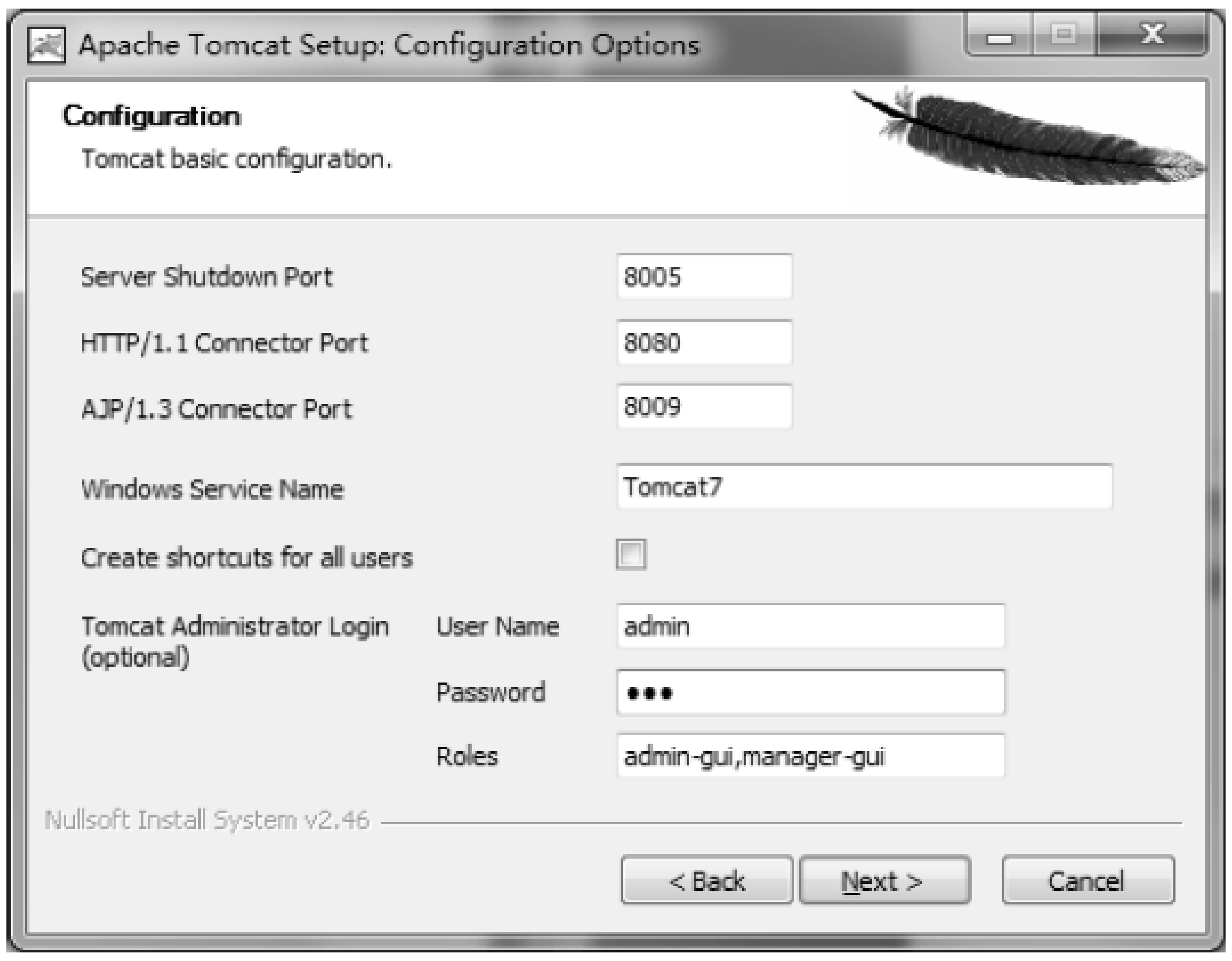


图 1.23　设置端口号和用户名及密码

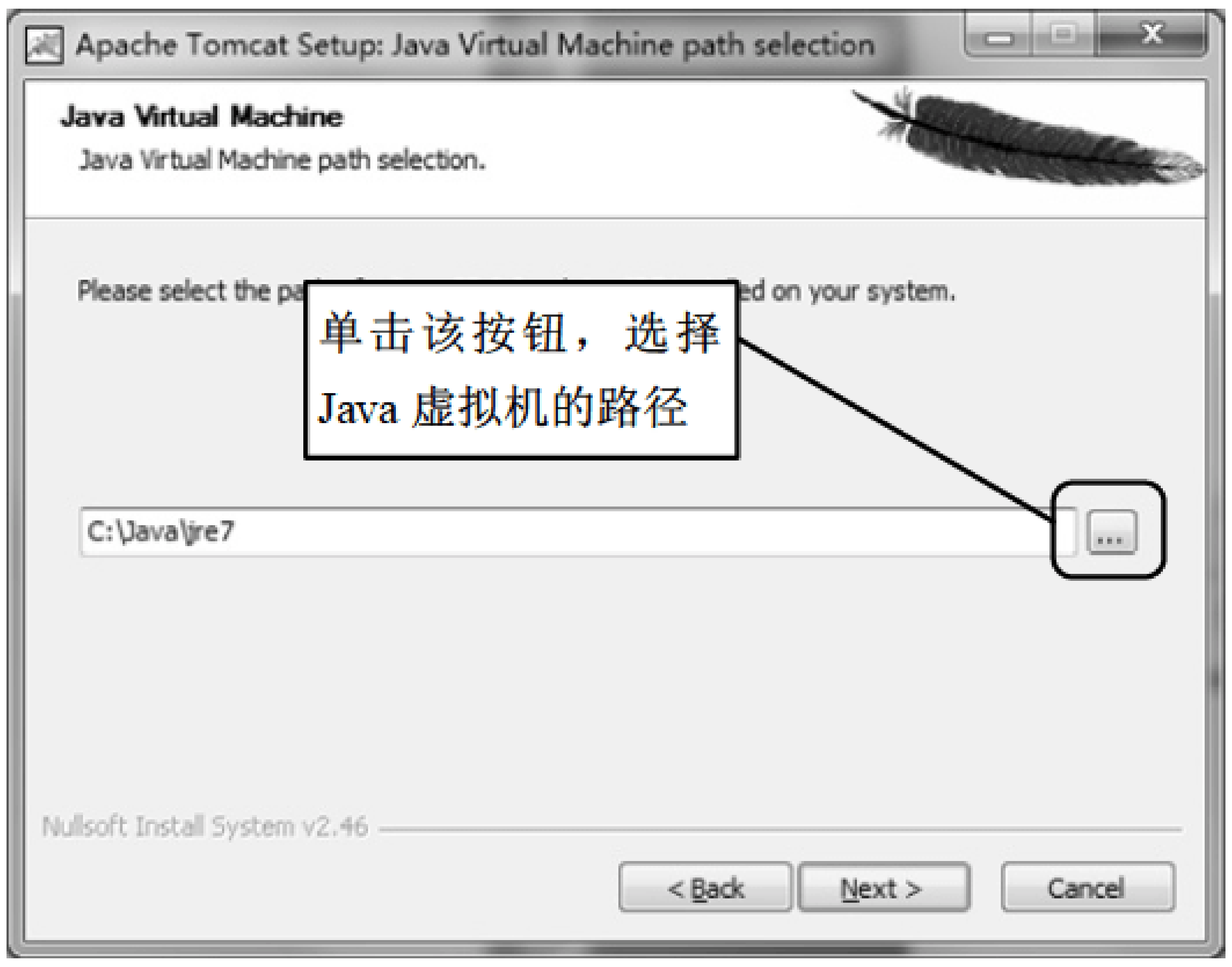


图 1.24　选择 Java 虚拟机路径

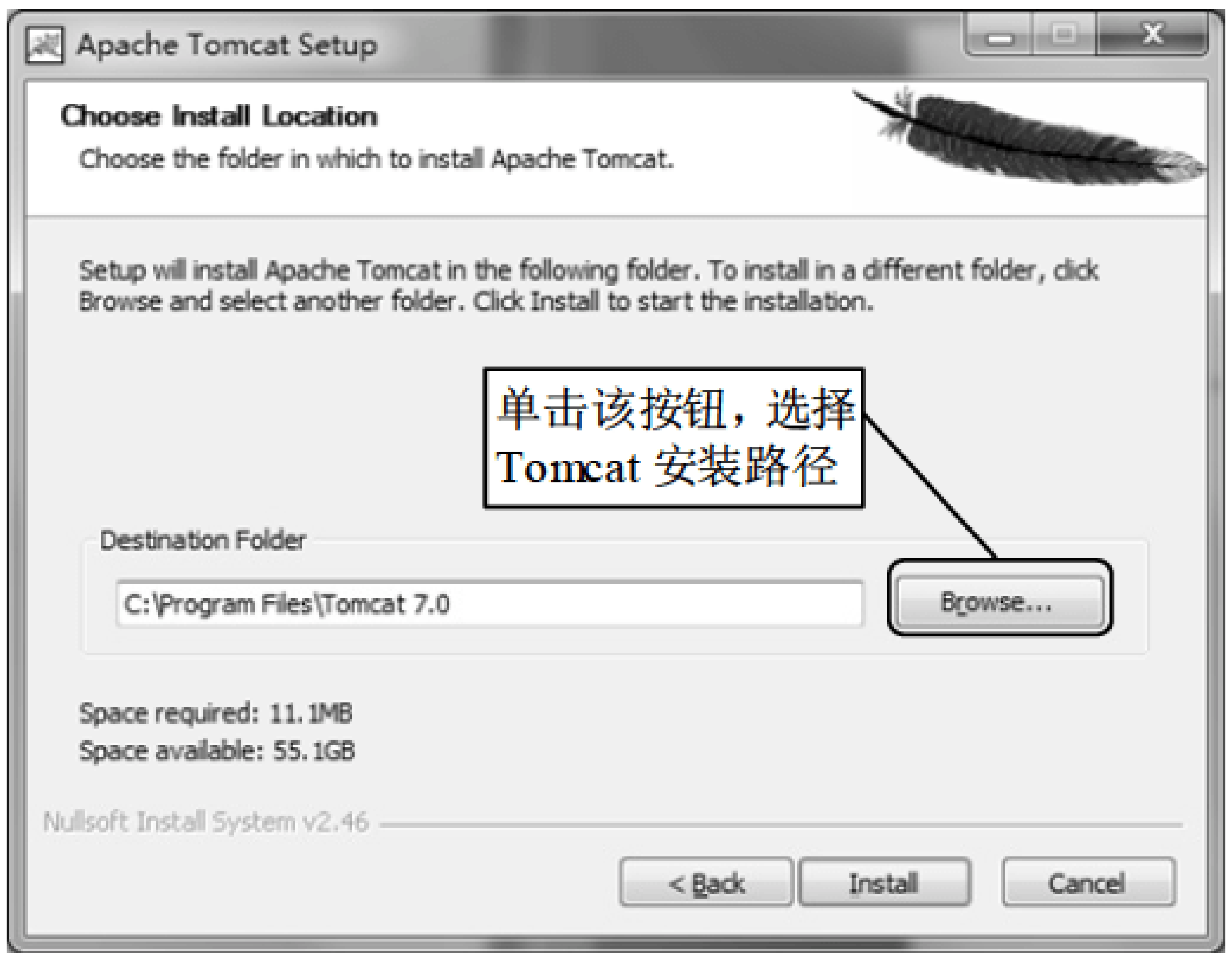


图 1.25　更改 Tomcat 的安装路径

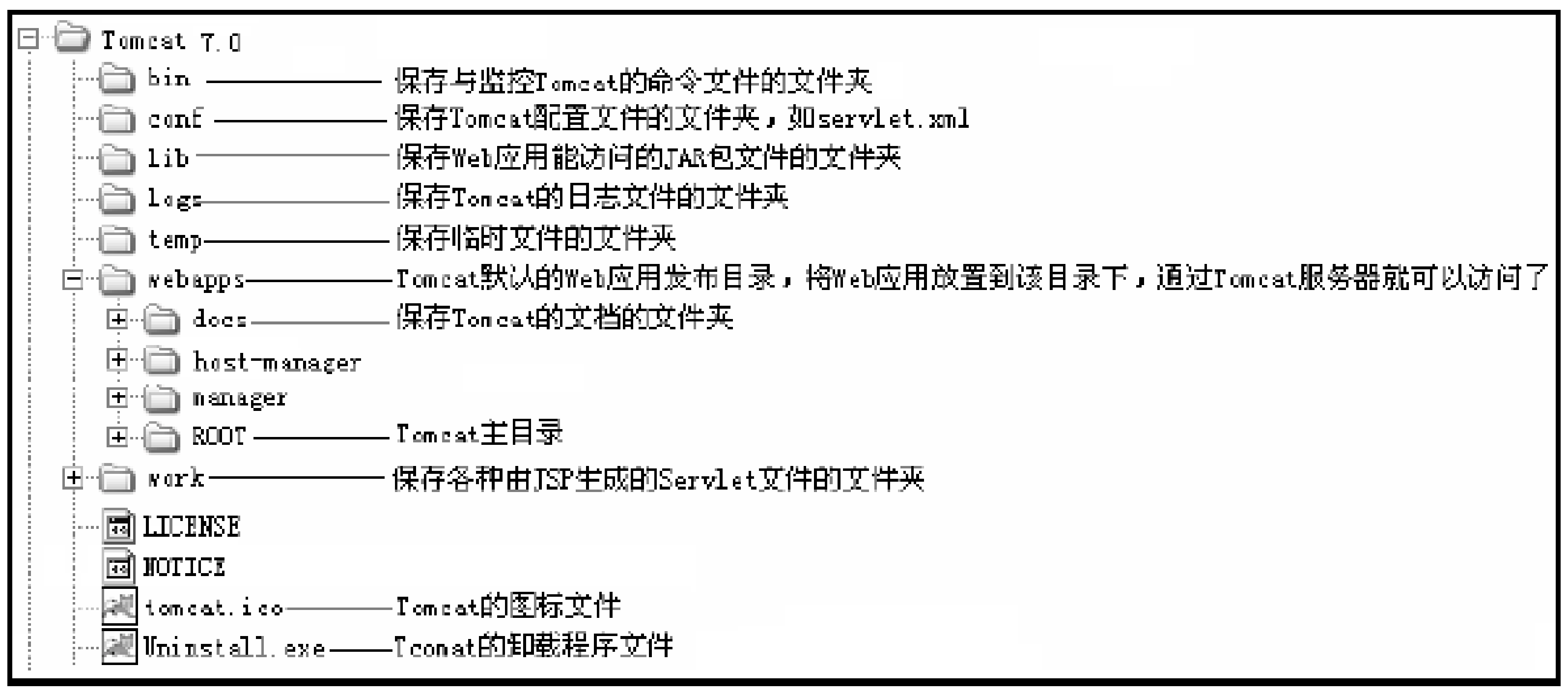


图 1.26　Tomcat 的目录结构

实例 008	启动 Tomcat 并测试 光盘位置：光盘\MR\01\008	初级 实用指数：★

实例说明

熟悉 Tomcat 服务器的内部结构后，可以启动 Tomcat 服务器来测试 Tomcat 是否能够成功运行。

设计过程

（1）启动 Tomcat。选择“开始”/“所有程序”/Apache Tomcat 7.0 Tomcat 7/Monitor Tomcat 命令，在任务栏右侧的系统托盘中将出现图标，在该图标上单击鼠标右键，在打开的快捷菜单中选择 Start service 命令，启动 Tomcat。

（2）打开 IE 浏览器，在地址栏中输入地址“http://localhost:8080”访问 Tomcat 服务器，若出现如图 1.27 所示的页面，则表示 Tomcat 安装成功。

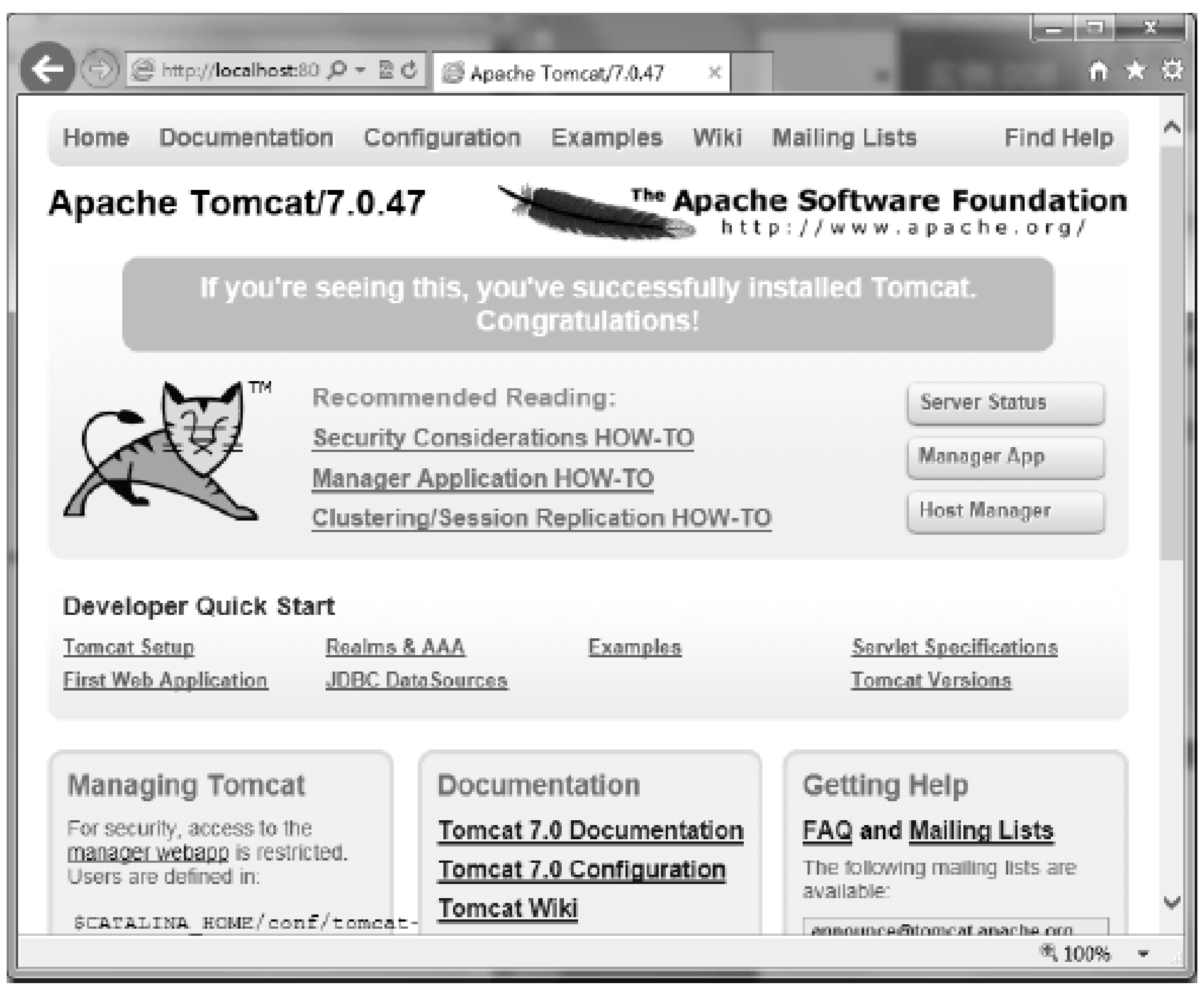


图 1.27　Tomcat 的启动界面

实例 009 通过 Eclipse 部署与发布 Web 应用

光盘位置：光盘\MR\01\009

初级

实用指数：★

实例说明

本实例介绍如何在 Eclipse 中部署 Web 应用，以及在 Eclipse 中启动 Tomcat 服务器的具体方法。

设计过程

（1）打开 Eclipse，在菜单中创建一个动态 Web 项目，如图 1.28 所示。

（2）选择 Dynamic Web Project 命令后，打开 New Dynamic Web Project 窗口，在 Project name 文本框中输入项目名称，如图 1.29 所示。

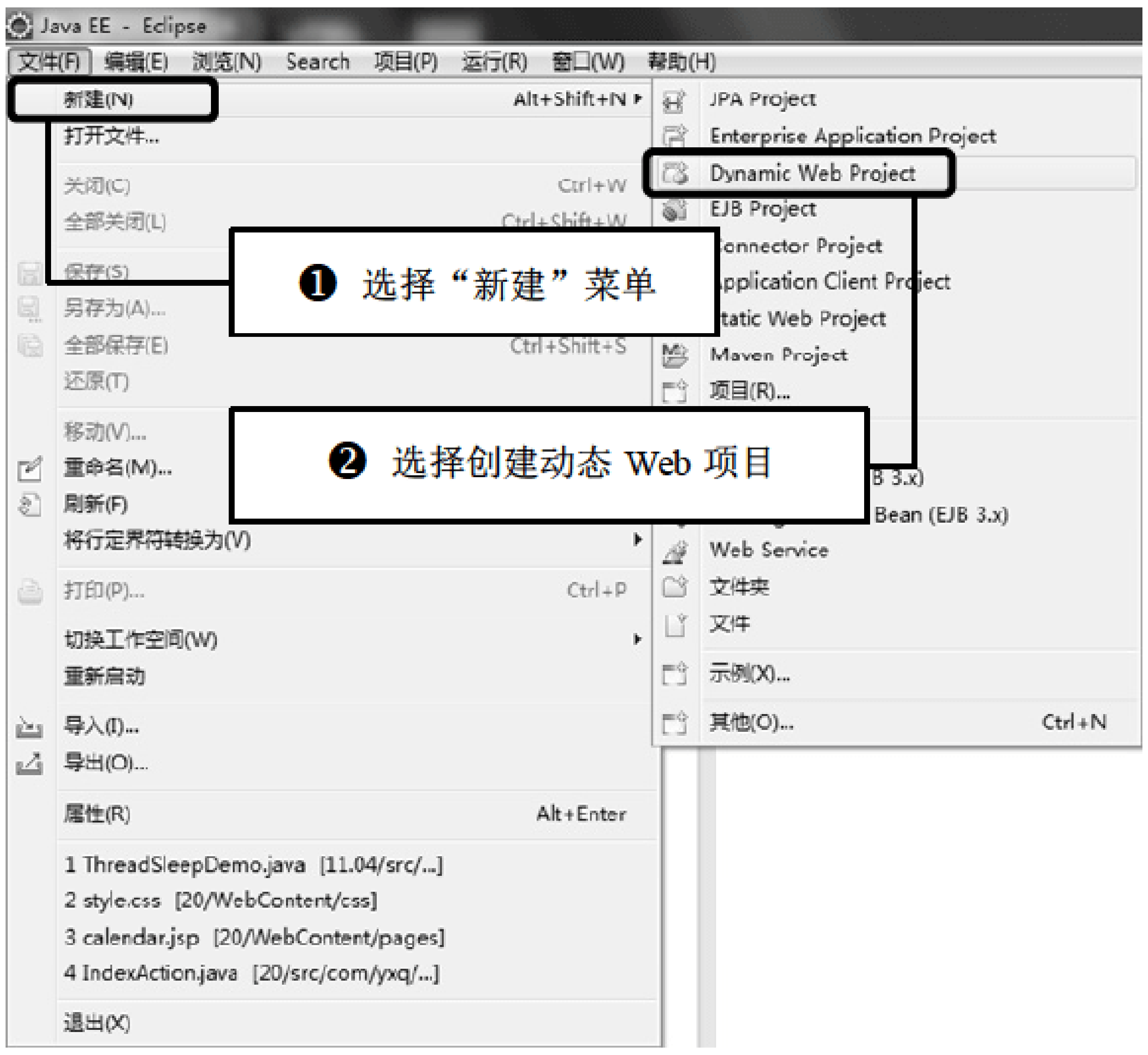


图 1.28　创建动态 Web 项目

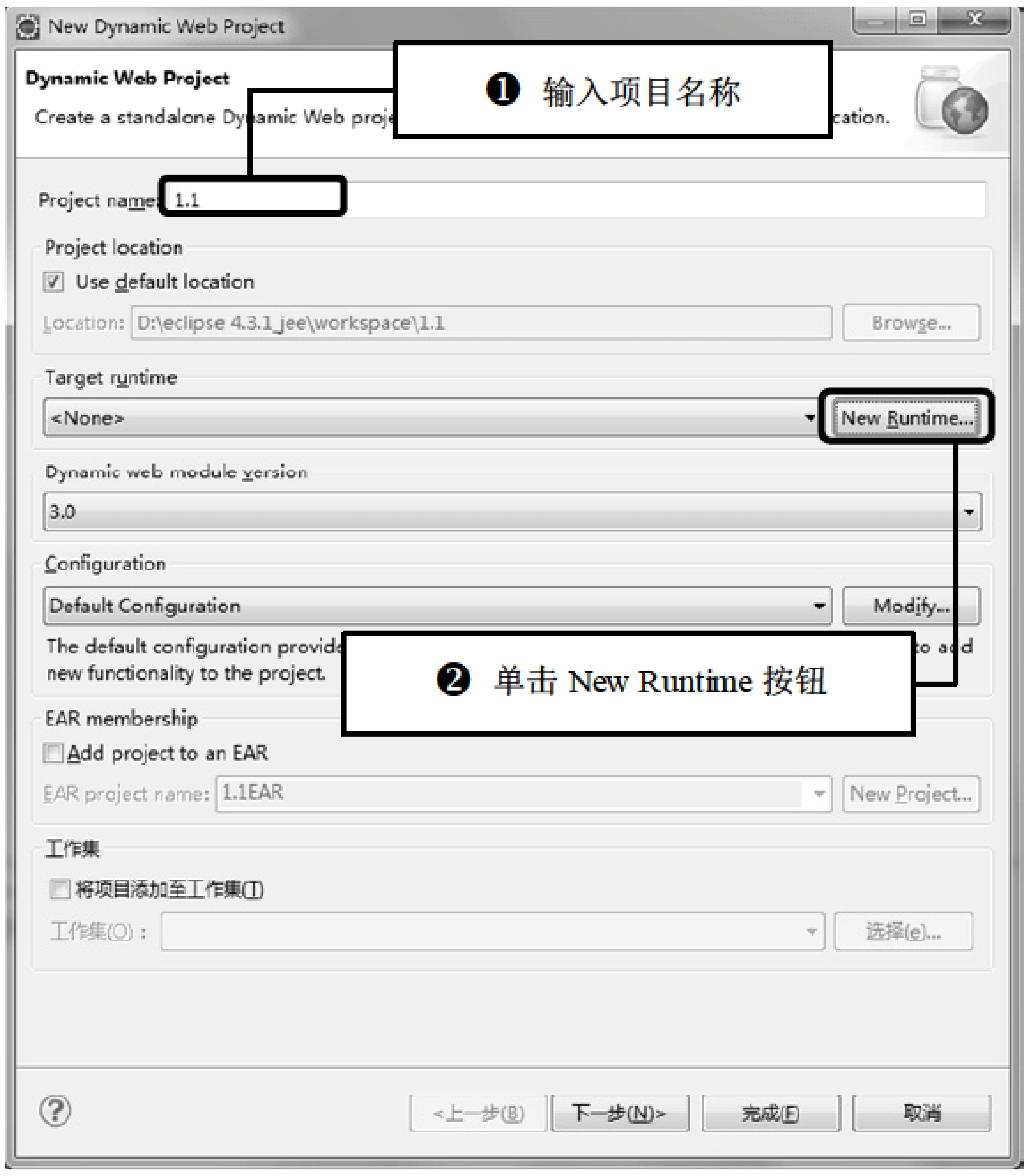


图 1.29　New Dynamic Web Project 窗口

（3）单击 New Runtime 按钮，新建一个运行时服务器，将弹出如图 1.30 所示的对话框。在该对话框中选择 Apache/Apache Tomcat v7.0 选项，单击“下一步”按钮，打开指定安装目录窗口，如图 1.31 所示。

说明：由于前面下载的是 Tomcat 7.0.47 版本，所以此处选择 Apache Tomcat v7.0 选项，与下载的版本相对应。

（4）单击 Browse 按钮，显示“浏览文件夹”对话框，在该对话框中选择 Tomcat 的根目录，如图 1.32 所示，单击“确定”按钮，返回到指定安装目录对话框，单击“完成”按钮，完成运行时服务器的创建，返回到 New Dynamic Web Project 窗口中。

说明：如果在 Eclipse 中，已经创建过运行时服务器，那么上面的步骤（3）和（4）可以省略。

（5）这时，在 Target runtime 下拉列表中将自动选择新创建的运行时服务器。单击“完成”按钮，项目创建成功。此时，可以在 Eclipse 的项目资源管理器中查看到已经创建的项目“1.1”，如图 1.33 所示。

（6）在 Eclipse 的项目资源管理器中，选中 WebContent 文件夹并单击鼠标右键，在弹出的快捷菜单中选择“新建”/JSP File 命令，打开 New JSP File 窗口。在该窗口的“文件名”文本框中输入文件名 index.jsp，其他采

用默认，如图 1.34 所示。

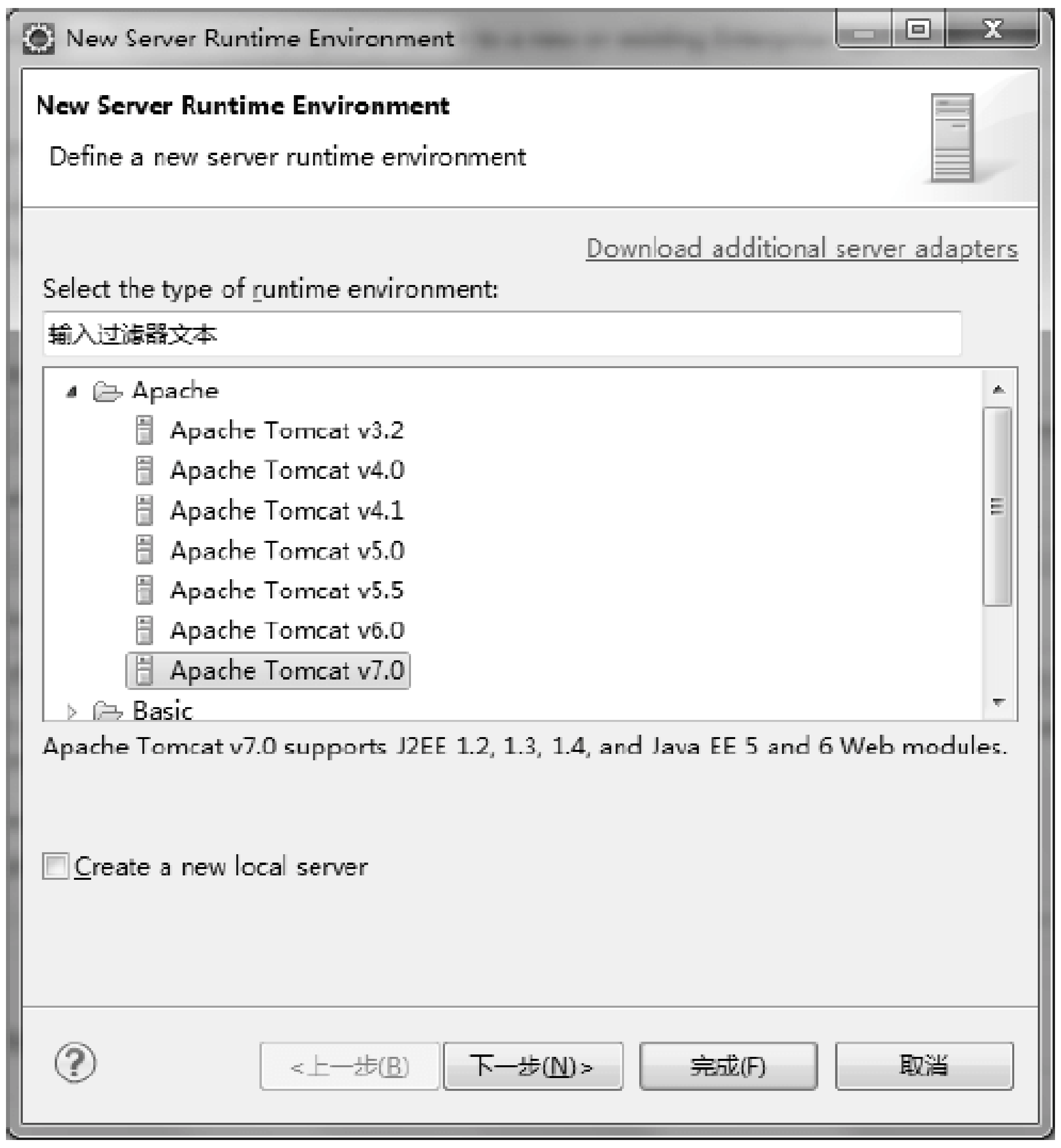


图 1.30　新建服务器运行时窗口



图 1.31　指定安装目录



图 1.32　“浏览文件夹”对话框

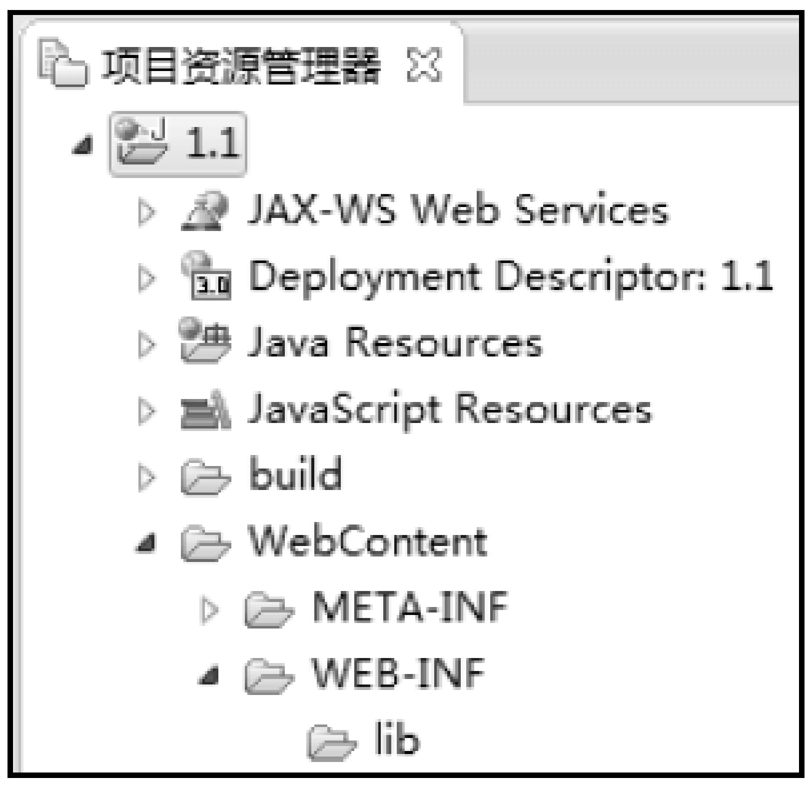


图 1.33　项目创建成功



图 1.34　创建 index.jsp 页面

（7）单击“完成”按钮，在 Eclipse 中打开 index.jsp 页面的代码窗口，在此页面中编写输出“欢迎来到明日图书网”的内容，代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Hello Welcome to MR</title>
        </head>
        <body>
                <center>欢迎来到明日图书网</center>
        </body>
</html>
```

（8）在项目名称节点上单击鼠标右键，在弹出的快捷菜单中选择“运行方式”/Run on Server 命令，即可启动 Tomcat 服务器运行本项目，运行结果如图 1.35 所示。



图 1.35　运行结果

实例 010　修改 Tomcat 服务器的端口号

初级

实用指数：★★

实例说明

本实例主要介绍修改 Tomcat 端口号的方法，以及当端口号发生冲突时如何解决。

关键技术

Connector 子元素下的 port 是设置服务器端口，而 connection Timeout 则是服务器连接超时，单位为毫秒。

设计过程

（1）使用记事本打开 Tomcat 安装目录下的 conf 文件夹下的 servlet.xml 文件。

（2）在 servlet.xml 文件中找到以下代码：

```
<Connector port="8080" protocol="HTTP/1.1"
            connectionTimeout="20000"
            redirectPort="8443" />
```

（3）将上面代码中的 port="8080"修改为 port="8081"，即可将 Tomcat 的默认端口设置为 8081。在修改端口时，应避免与公用端口冲突。建议采用默认的 8080 端口，不要修改，除非 8080 端口被其他程序所占用。

（4）修改成功后，为了使新设置的端口生效，还需要重新启动 Tomcat 服务器。

（5）将 Tomcat 端口号更改为 8081 后，重新启动的界面如图 1.36 所示。

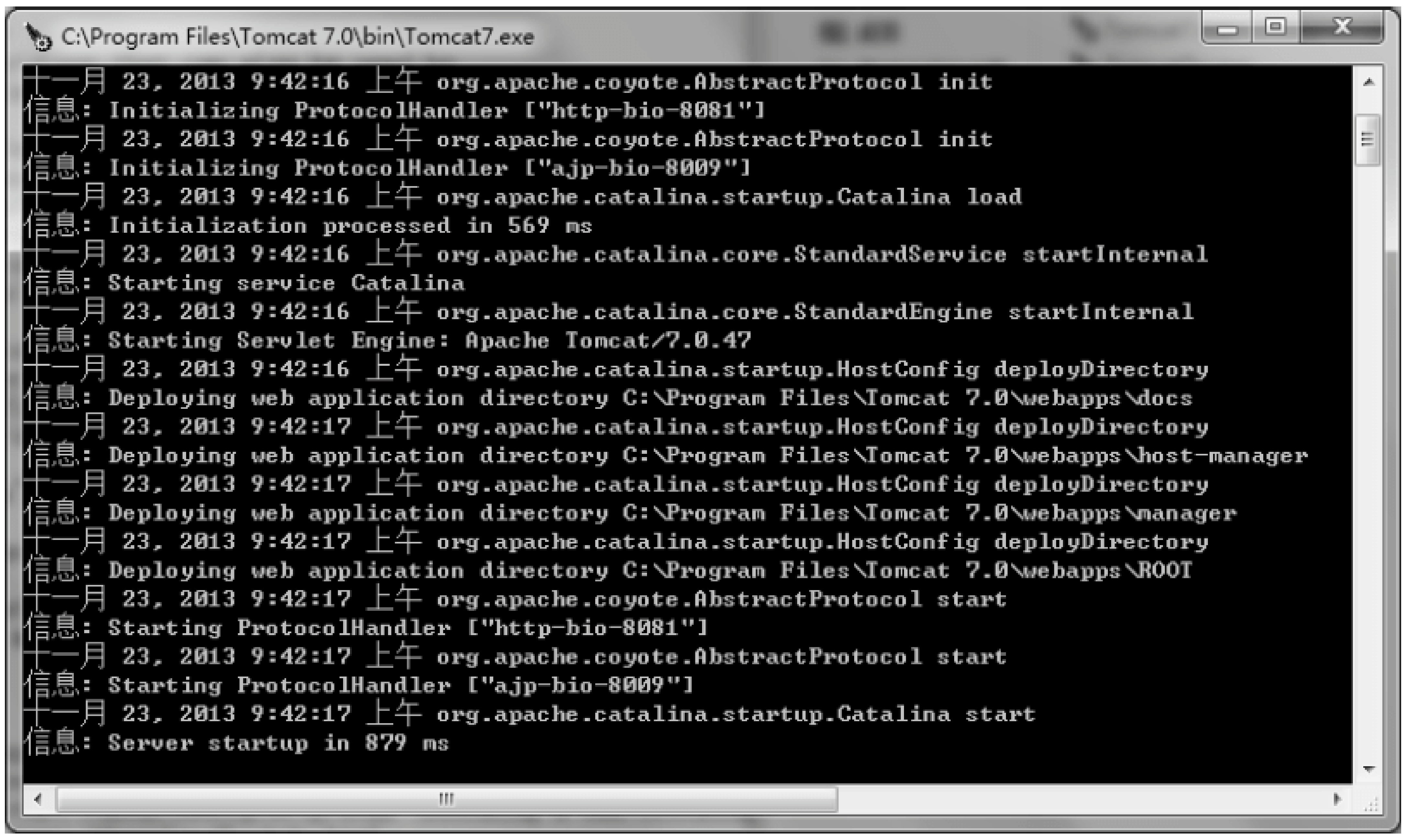


图 1.36　更改端口号后重新启动的界面

实例 011　配置 Tomcat 的虚拟主机

初级

实用指数：★★

实例说明

本实例介绍如何配置 Tomcat 的虚拟主机。

关键技术

关于 server.xml 中 Host 这个元素，只有在设置虚拟主机时才需要修改。虚拟主机是一种在一个 Web 服务器上服务多个域名的机制，对每个域名而言，都好像独享了整个主机。实际上，大多数的小型商务网站都是采用虚拟主机实现的，这主要是因为虚拟主机能直接连接到 Internet 并提供相应的带宽，以保障合理的访问响应速度。另外，虚拟主机还能提供一个稳定的固定 IP。

设计过程

（1）打开 Tomcat 根目录下的 conf 文件夹，然后用记事本打开 server.xml 文件，并在其中添加如下代码：

```
<host name="www.example.com" appBase="/home/example/webapp">
<Context path="" docBase="."/> </host>
```

（2）Tomcat 的 server.xml 文件在初始状态下，只包括一个虚拟主机，但是它很容易被扩充到支持多个虚拟主机。上面展示的是一个简单的 server.xml 版本，其中粗体部分就是用于添加一个虚拟主机。每一个 Host 元素必须包括一个或多个 context 元素，所包含的 context 元素中必须有一个是默认的 context，这个默认的 context 的显示路径应该为空（例如，path= “ ”）。

实例 012　在 Tomcat 下如何手动部署 Web 应用

初级

实用指数：★★

实例说明

通过对 Tomcat 目录的了解可知，webappas 文件夹是存放工程包的位置。本实例主要介绍如何手动部署 Web 应用。

关键技术

这种 context 片段提供了一种便利的方法来部署 Web 应用，不需要编辑 server.xml，除非想改变默认的部署特性，在安装一个新的 Web 应用时不需要重新启动 Tomcat。

设计过程

（1）复制 war 文件或者 Web 应用文件夹（包括 Web 下所有内容）到$CATALINA_BASE\webapps 目录下。

（2）为 Web 服务建立一个只包括 context 内容的 XML 片段文件，并把该文件放到$CATALINA_BASE\webapps 目录下，这个 Web 应用文件本身可以存储在硬盘的任何地方。

（3）部署 Web 应用文件的另一种方式是写一个 Context XML 片段文件，然后把该文件复制到$CATALINA_BASE\webapps 目录下。一个 Context 片段并不是一个完整的 XML 文件，只是一个 context 元素，以及对该应用文件的相应描述。这种片段文件就像是从 server.xml 中提取出来的 context 元素一样，所以这种片段被命名为“context 片段”。如果想部署一个名叫 MyWeb.war 的应用文件，该应用文件使用 realm 作为访问控制方式，可以使用下面这个片段并添加以下代码：

```
<context path="/demo" docBase="webapps/MyWeb.war" debug="0" privileged="true">
```

```
<Realm className="org.apache.catalina.realm.UserDatabaseRealm"    resourceName="UserDatabase"/>
</context>
```

实例 013 Tomcat 如何制定主机访问

初级

实用指数：★★

实例说明

有时需要限制对 Tomcat Web 应用的访问，如只有指定的主机或 IP 地址可以访问指定的应用。这样一来，就只有那些指定的客户端可以访问服务的内容。本实例将介绍如何在 Tomcat 下制定主机访问。

关键技术

Tomcat 提供了两个参数供用户配置，即 RemoteHostValve 和 RemoteAddrValve。通过配置这两个参数，可以过滤来自请求的主机或 IP 地址，并允许或拒绝哪些主机/IP。

设计过程

（1）在 Apache 的 httpd 文件中有对每个目录的允许/拒绝指定，如可以把 Admin Web application 设置成只允许本地访问，代码如下：

```
<context path="/path/to/secret_files" ...>
        <Valve className="org.apache.catalina.valves.RemoteAddrValve" allow="127.0.0.1" deny=""/>
</context>
```

（2）如果没有给出允许主机的指定，那么与拒绝主机匹配的主机就会被拒绝，除此之外的都是允许的。与之类似，如果没有给出拒绝主机的指定，那么与允许主机匹配的主机就会被允许，除此之外的都是拒绝的。

实例 014 Tomcat 如何添加管理员

初级

实用指数：★

实例说明

为 Tomcat 添加管理员，在 Tomcat 界面化管理平台中可以看到所有加载的工程包，以及运行的平台，还可以对项目进行管理，如删除和添加，来实现对服务器的维护与更新。

设计过程

（1）打开 Tomcat 7.0 目录下的 conf 文件夹中的 tomcat-users.xml 文件，添加以下代码：

```
<user username="zm" password="zm" roles="admin-gui,manager-gui"/>
```

（2）重新启动 Tomcat。在 IE 浏览器的地址栏中输入“http://localhost:8080/manager”，在打开的界面中输入刚刚添加的账号和密码，如图 1.37 所示。

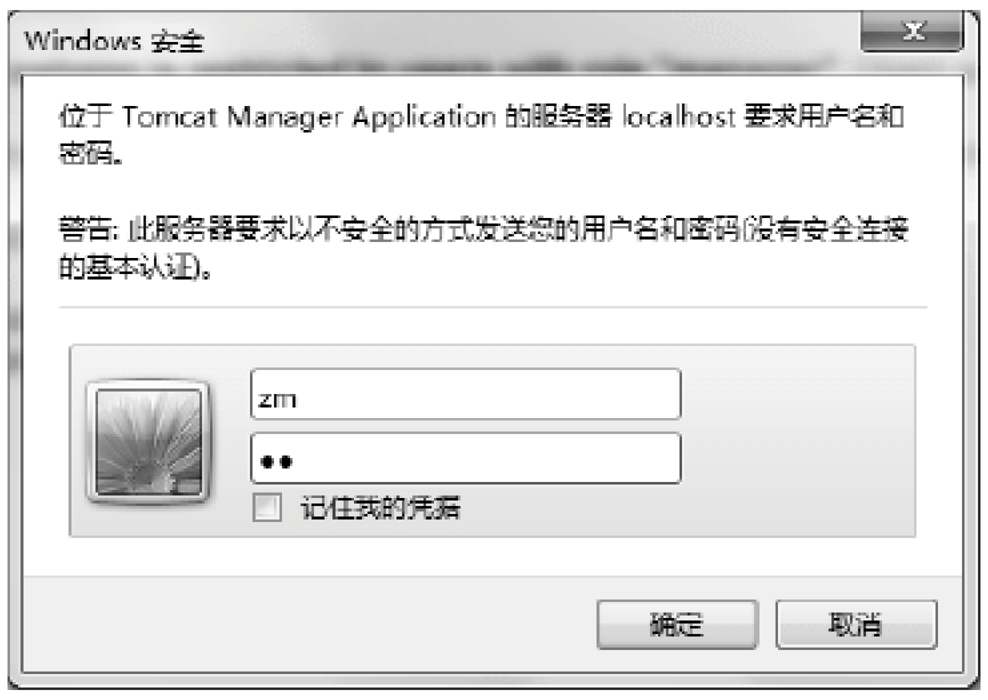


图 1.37　登录 Tomcat 界面

（3）登录之后即可进入到如图 1.38 所示的 Tomcat Web 应用管理页面。在该页面中可以对项目进行添加与删除管理。

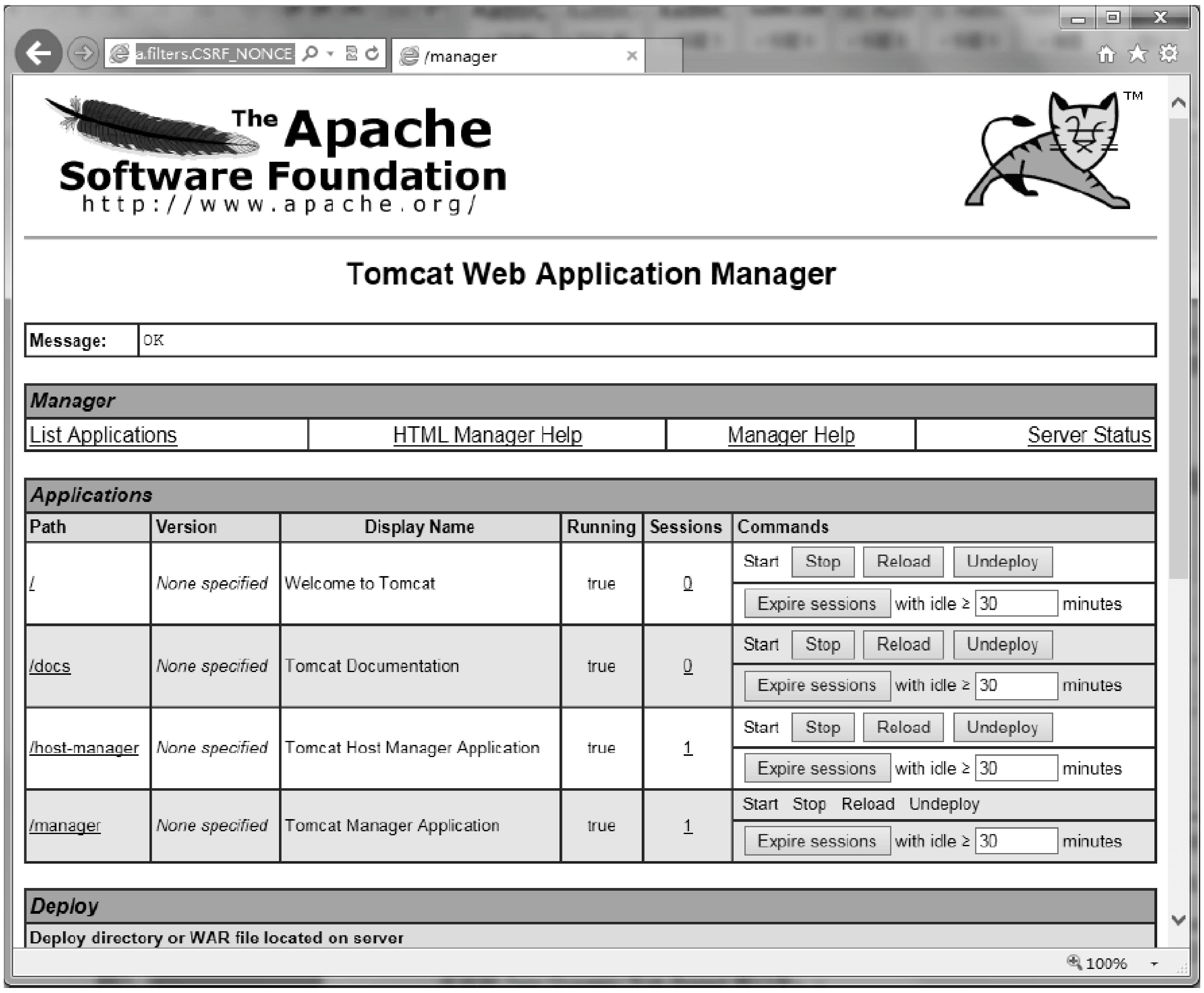


图 1.38　登录 Tomcat 之后的界面

实例 015　Tomcat 常用的优化技巧

初级

实用指数：★

实例说明

本实例介绍的是如何优化 Tomcat 服务器，如果用户并发量小，系统可能不会出问题，但是并发量大时，系统反应速度迅速下降，由于不了解原因拼命在自己的应用中寻找问题，从而浪费了宝贵的时间。下面来看看 Tomcat 是如何优化的。

设计过程

（1）屏蔽 DNS 查询

Web 应用程序可以通过 Web 容器提供的 getRemoteHost()方法获得访问 Web 应用客户的 IP 地址和名称，但是这样会消耗 Web 容器的资源，并且还需要通过 IP 地址和 DNS 服务器反查用户的名字。因此当系统上线时，可以将这个属性关闭，从而减少资源消耗，那么 Web 应用也就只能记录下 IP 地址。修改的属性是 enableLoopups="false"。

（2）调整线程数

Tomcat 通过线程池来为用户访问提供响应，对于上线的系统初步估计用户并发数量后，再调整线程池容量。例如，用户并发数量在 100 左右时，可以设置 minProcessors="100"，maxProcessors="100"。将最大和最小设置为一样后，线程池不会再释放空闲的线程，当用户访问突然增加时，不需要再消耗系统资源去创建新的线程。

（3）调整最大连接数

这个其实最复杂，即使用户并发量大，但是系统反应速度快，也没必要把这个值设置太高，高了系统需要消耗大量的资源去切换线程，但是如果设置太低也会造成应用无法满足用户并发需要。因此设置这个最好能够结合整个系统的跟踪与调优，使系统达到最好的平稳状态，一般设置为 maxProcessors 的 1.5 倍即可。

（4）调整网络超时

主要是 HTTP 协议也有个连接过程，客户端连接到服务器上后，如果长时间没有得到处理就会被释放。如果服务器处理速度较慢，但是希望每个用户都能得到有效处理，或者网络环境不好，需要保证用户不会因为超时中断，也可以把时间加长。但是一般设置成 connectionTimeout="30000"即可。太长对系统来说价值不大，反而会浪费系统资源在无谓的长连接上。

（5）具体修改如下

- ❑ minProcessors：最小空闲连接线程数，用于提高系统处理性能，默认值为 10。
- ❑ maxProcessors：最大连接线程数，即并发处理的最大请求数，默认值为 75。
- ❑ acceptCount：允许的最大连接数，应大于等于 maxProcessors，默认值为 100。
- ❑ enableLookups：是否反查域名，取值为 true 或 false。为了提高处理能力，应设置为 false。
- ❑ connectionTimeout：网络连接超时，单位为毫秒。设置为 0 表示永不超时，但这样设置存在隐患，通常可设置为 20000 毫秒。

1.3 Linux 系统配置 JDK 与 Tomcat 服务器

实例 016 在 Linux 系统下安装配置 JDK

高级

实用指数：★

实例说明

本实例介绍的是如何在 Linux 系统下安装配置 JDK。具体是通过 shell 命令来实现 JDK 的安装以及配置，应用的是 Linux 5.4 服务器版本。

说明：*在阅读本实例之前，需要读者对 Linux 系统环境有一定的了解；否则，可以直接越过本实例。*

设计过程

（1）在安装 JDK 之前，首先访问 JDK 的官方网站（http://java.sun.com/j2se/1.4.2/download.html），下载一个 Linux Platform 的 JDK，建议下载 RPM，自解压格式的（RPM in self-extracting file，j2sdk-1_4_2_06-linux-i586-rpm.bin）。

（2）上传到 Linux 服务器上，在 shell 下执行以下命令：

```
[root@LinuxServer rpm]# chmod 755 j2sdk-1_4_2_06-linux-i586-rpm.bin
[root@LinuxServer rpm]# ./j2sdk-1_4_2_06-linux-i586-rpm.bin
```

（3）执行以上命令之后，会有一段 Sun 的安装注册协议条款，按几次空格键，当询问是否同意安装协议时，再输入“yes”即可，代码如下：

```
Sun Microsystems, Inc.
Binary Code License Agreement
for the
JAVATM 2 SOFTWARE DEVELOPMENT KIT (J2SDK), STANDARD
EDITION, VERSION 1.4.2_X
...
Do you agree to the above license terms? [yes or no]yes
Unpacking...
Checksumming...
0
```

```
0
Extracting...
UnZipSFX 5.40 of 28 November 1998, by Info-ZIP (Zip-Bugs@lists.wku.edu).
 inflating: j2sdk-1_4_2_06-linux-i586.rpm
Done.
```

（4）程序会自动生成一个 j2sdk-1_4_2_06-linux-i586.rpm 文件，这是主程序包。下面进行安装，安装命令如下：

```
[root@LinuxServer rpm]#rpm -ivh j2sdk-1_4_2_06-linux-i586.rpm
Preparing...          ########################################### [100%]
1:j2sdk          ########################################### [100%]
```

（5）安装完 JDK 之后，还需要配置 JDK 的环境变量，这里用 export 命令直接在 shell 下设置，具体命令如下：

```
[root@LinuxServer rpm]# export JAVA_HOME=/usr/java/j2sdk1.4.2_06
[root@LinuxServer rpm]# export
CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
[root@LinuxServer rpm]# export PATH=$PATH:$JAVA_HOME/bin
```

（6）当然，在步骤（5）中设置环境变量是可以生效的，但是只对当前 shell 生效。如果从另外一个 shell 登录，将不能使用刚才设置的变量。所以最好的方法还是修改.bashrc 文件，具体修改命令如下：

```
[root@LinuxServer rpm]#vi .bashrc
set JAVA_HOME=/usr/java/j2sdk1.4.2_06
export JAVA_HOME
set PATH=$PATH:$JAVA_HOME/bin
export PATH
set CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
export CLASSPATH
```

（7）当然也可以通过更改/etc/profile 来实现，不过不推荐这么做，因为这样设置将对所有用户的 shell 都生效，从而对系统安全会产生影响。下面来验证一下变量设置是否生效（在验证前先 logout，再重新登录），代码如下：

```
[root@LinuxServer rpm]# echo $JAVA_HOME
/usr/java/j2sdk1.4.2_06/
[root@LinuxServer rpm]# echo $CLASSPATH
/usr/java/j2sdk1.4.2_06/lib/dt.jar:/usr/java/j2sdk1.4.2_06/lib/tools.jar
[root@LinuxServer rpm]# echo $PATH
/usr/java/j2sdk1.4.2_06/bin/:/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/sbin:/usr/local/bin:
/sbin:/bin:/usr/sbin:/usr/bin:/usr/X11R6/bin:/root/bin
[root@LinuxServer rpm]# JAVA-version
JAVA version "1.4.2_06"
JAVA(TM) 2 Runtime Environment, Standard Edition (build 1.4.2_06-b03)
JAVA HotSpot(TM) Client VM (build 1.4.2_06-b03, mixed mode)
```

（8）在安装配置完 JDK 之后，看看 JDK 是否能正常工作。下面来写一个测试文件 test.java，代码如下：

```
[root@LinuxServer rpm]#vi test.java
class test {
      public static void main(String[] args)
      {
           System.out.println("Hello World!");
      }
}
```

（9）保存退出。下面来编译、执行，具体命令如下：

```
[root@LinuxServer text]# javac test.java
[root@LinuxServer text]# JAVA test
```

当运行 java test 命令之后，将会打印出字符串“Hello World！”。

（10）如果要使某个用户具有运行 Java 命令的权限，只要修改其 bash 初始化文件即可。如要授予用户 long ware 运行 Java 命令的权限，执行以下命令：

```
[root@LinuxServer root]# vi /home/longware/.bashrc
set JAVA_HOME=/usr/java/j2sdk1.4.2_06
export JAVA_HOME
set PATH=$PATH:$JAVA_HOME/bin
export PATH
set CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
export CLASSPATH
```

（11）至此，Linux 系统上的 JDK 安装完毕。

实例 017　在 Linux 系统下安装配置 Tomcat

高级

实用指数：★

实例说明

实例 016 介绍了如何在 Linux 系统下安装配置 JDK，在此之后，还有一个必须要完成的任务，就是安装配置 Tomcat 服务器。本实例介绍的是如何在 Linux 系统下安装配置 Tomcat 服务器。

说明：在阅读本实例之前，需要读者对 Linux 系统环境有一定的了解；否则，可以直接越过本实例。

设计过程

（1）首先要把 apache-tomcat-6.0.20.tar.gz 上传到 Linux，然后开始执行下面的命令：

```
[root@JavaServer ~]# cd download/
[root@JavaServer download]# ll
total 84208
-rw-r--r-- 1 root root  5998298 Aug 21 11:36 apache-tomcat-6.0.20.tar.gz
-rwxr-xr-x 1 root root 80129716 Aug 21 10:11 jdk-6u13-linux-i586.bin
```

（2）使用 tar-zxvf apache-tomcat-6.0.20.tar.gz 解压文件，屏幕将显示解压信息，命令如下：

```
[root@JavaServer download]# tar -zxvf apache-tomcat-6.0.20.tar.gz
```

（3）使用 ll 显示出 Tomcat 文件夹 apache-tomcat-6.0.20，命令如下：

```
[root@JavaServer download]# ll
total 84212
drwxr-xr-x 9 root root     4096 Jan  8 12:11 apache-tomcat-6.0.20
-rw-r--r-- 1 root root  5998298 Aug 21 11:36 apache-tomcat-6.0.20.tar.gz
-rwxr-xr-x 1 root root 80129716 Aug 21 10:11 jdk-6u13-linux-i586.bin
```

（4）使用 mv apache-tomcat-6.0.20/usr/local 把文件夹移动到其 usr 目录下，具体命令如下：

```
[root@JavaServer download]# mv apache-tomcat-6.0.20 /usr/local/
```

（5）使用 mv apache-tomcat-6.0.20 tomcat6 修改目录名为 tomcat6，具体命令如下：

```
[root@JavaServer download]# cd /usr/local
[root@JavaServer local]# ll
total 76
drwxr-xr-x 9 root root 4096 Jan  8 12:11 apache-tomcat-6.0.20
drwxr-xr-x 2 root root 4096 Mar 10  2009 bin
drwxr-xr-x 2 root root 4096 Mar 10  2009 etc
drwxr-xr-x 2 root root 4096 Mar 10  2009 games
drwxr-xr-x 2 root root 4096 Mar 10  2009 include
drwxr-xr-x 2 root root 4096 Mar 10  2009 lib
drwxr-xr-x 2 root root 4096 Mar 10  2009 libexec
drwxr-xr-x 2 root root 4096 Mar 10  2009 sbin
drwxr-xr-x 4 root root 4096 Aug 30 17:32 share
drwxr-xr-x 2 root root 4096 Mar 10  2009 src
[root@JavaServer local]# mv apache-tomcat-6.0.20/ tomcat6
```

（6）进入到 tomcat6/bin 目录下，startup.sh 和 catalina.sh 这两个文件能启动 Tomcat，命令如下：

```
[root@JavaServer tomcat6]# cd bin
```

（7）使用./startup.sh 启动服务，命令如下：

```
[root@JavaServer bin]# ./startup.sh
Using CATALINA_BASE:   /usr/local/tomcat6
Using CATALINA_HOME:   /usr/local/tomcat6
Using CATALINA_TMPDIR: /usr/local/tomcat6/temp
Using JRE_HOME:       /usr/share/jdk1.6.0_13
```

（8）如果没什么问题，则 Tomcat 安装成功，但可能会出现类似如下内容的错误：

```
"[root@localhost bin]# ./startup.sh
Neither the JAVA_HOME nor the JRE_HOME environment variable is defined
At least one of these environment variable is needed to run this program"
```

（9）如果出现以上错误，是由于找不到 java_home 的路径造成的，笔者在安装时也出现过这些问题。这个问题的解决方法就是，按照自己在配置 JDK 的环境变量时的 java_home 的路径去制定好 java_home 的路径。使用 JAVA_HOME=/usr/java/jdk1.6.0_16 和 export JAVA_HOME 设置变量，再执行./startup.sh，显示如下信息：

```
"Using CATALINA_BASE:   /usr/tomcat5
Using CATALINA_HOME:   /usr/tomcat5
Using CATALINA_TMPDIR: /usr/tomcat5/temp
Using JRE_HOME:        /usr/java/jdk1.6.0_16
Using CLASSPATH:       /usr/tomcat5/bin/bootstrap.jar"
```

（10）使用 ps -ef |grep tomcat 可以显示 Tomcat 已启动，或者应用以下命令：

```
[root@JavaServer bin]# netstat -ltnp
```

（11）Tomcat 服务器默认为 8080 端口，打开浏览器，测试本地是否能上网，如果能访问 Tomcat 的首页界面，说明 Tomcat 服务器已安装成功。

（12）如果不能访问 Tomcat 的首页界面，本地上网则先关闭防火墙 service iptables stop，应用./shutdown.sh 命令结束 Tomcat 服务，也可以使用 kill PID 命令杀死 Tomcat 进程。下面把 Tomcat 添加到开机启动，修改/etc/rc.local 文件，在文件的最后添加如下内容：

```
JAVA_HOME=/usr/share/jdk1.6.0_13
export JAVA_HOME
/usr/local/tomcat6/bin/startup.sh
```

执行如下命令：

```
[root@JavaServer jdk1.6.0_13]# vi /etc/rc.local
```

然后执行如下命令：

```
touch /var/lock/subsys/local
JAVA_HOME=/usr/share/jdk1.6.0_13
export JAVA_HOME
/usr/local/tomcat6/bin/startup.sh

~
```

最后添加内容，保存退出，命令如下：

```
[root@JavaServer jdk1.6.0_13]# source /etc/rc.local
Using CATALINA_BASE:   /usr/local/tomcat6
Using CATALINA_HOME:   /usr/local/tomcat6
Using CATALINA_TMPDIR: /usr/local/tomcat6/temp
Using JRE_HOME:        /usr/share/jdk1.6.0_13
```

至此，Tomcat 开机即可启动。

第 2 章

Java 语言基础

- 基本语法
- 运算符
- 条件语句
- 循环控制
- 常用排序
- 算法应用

2.1　基 本 语 法

实例 018　输出错误信息与调试信息

光盘位置：光盘\MR\02\018

高级

实用指数：★★★☆

实例说明

程序开发中对于业务代码的部分功能需要配合调试信息以确定代码执行流程和数据的正确性，当程序出现严重问题时还要输出警告信息，这样可以在调试中完成程序开发。本实例将介绍如何输出调试信息与错误提示信息，实例运行结果如图 2.1 所示。

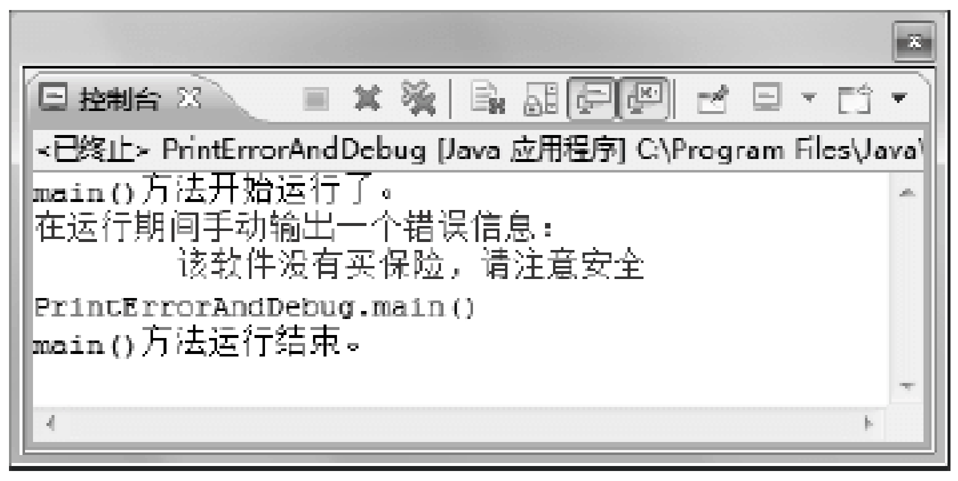


图 2.1　实例运行结果

关键技术

本实例使用 System 类中的 out 和 err 两个成员变量来完成调试信息与错误信息的输出，它们两个都是 System 的类变量，也就是说是使用 static 关键字修饰的。out 是标准调试信息的输出流，err 是标准错误信息输出流。本实例中调用了两个输出流通用的 println()方法来输出一行数据。该方法的声明如下：

```
public void println(String x)
```

参数说明

x：被输出到控制台的字符串。

设计过程

创建 PrintErrorAndDebug 类，并完成该类的 main()主方法，在该方法中分别输出调试信息与错误信息，关键代码如下：

```
public class PrintErrorAndDebug {
    public static void main(String[] args) {
        System.out.println("main()方法开始运行了。");
        //输出错误信息
        System.err.println("在运行期间手动输出一个错误信息：");
        System.err.println("\t 该软件没有买保险，请注意安全");
        System.out.println("PrintErrorAndDebug.main()");
        System.out.println("main()方法运行结束。");
    }
}
```

秘笈心法

System 类的 out 与 err 是两个类成员变量，不用创建 System 类的实例对象就可以直接使用。虽然都是标准输出流，但是应该灵活运用它们完成不同的信息输出。out 主要是输出调试信息的输出流，在 Eclipse 控制台中以黑色字体标识；而 err 是错误信息的标准输出流，用于输出紧急错误信息，所以在 Eclipse 控制台中以红色字体显示。

实例 019　从控制台接收输入字符

光盘位置：光盘\MR\02\019

高级

实用指数：★★★☆

实例说明

System 类除了 out 和 err 两个输出流之外，还有 in 输入流的实例对象作为类成员，它可以接收用户的输入。

本实例通过该输入流实现从控制台接收用户输入文本，并提示该文本的长度信息，实例运行结果如图 2.2 所示。

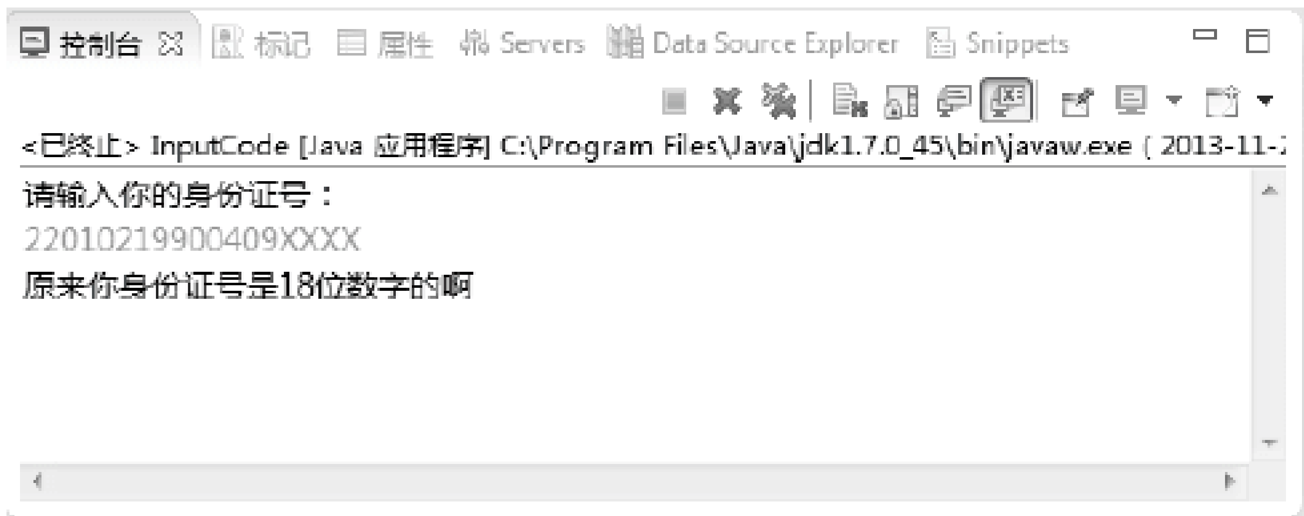


图 2.2　实例运行结果

关键技术

本实例的关键技术用到了 System 类的输入流，也就是类变量 in，它可以接收用户的输入信息，并且是标准的输入流实例对象。另外，Scanner 类是 Java 的扫描器类，它可以从输入流中读取指定类型的数据或字符串。本实例使用 Scanner 类封装了输入流对象，并使用 nextLine()方法从输入流中获取用户输入的整行文本字符串，该方法的声明如下：

```
public String nextLine()
```

该方法从扫描器封装的输入流中获取一行文本字符串作为方法的返回值。

设计过程

创建 InputCode 类，在该类的主方法中创建 Scanner 扫描器来封装 System 类的 in 输入流，然后提示用户输入身份证号码，并输出用户身份证号码的位数，关键代码如下：

```
public class InputCode {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);          //创建输入流扫描器
        System.out.println("请输入你的身份证号：");        //提示用户输入
        String line = scanner.nextLine();                  //获取用户输入的一行文本
        //打印对输入文本的描述
        System.out.println("原来你身份证号是" + line.length() + "位数字的啊");
    }
}
```

秘笈心法

InputStream 输入流以字节为单位来获取数据，而且需要复杂的判断并创建字节数组作为缓冲，最主要的是字节转换为字符时容易出现中文乱码的情况。所以对于字符数据的读取，应该使用扫描器进行封装，然后获取字符串类型的数据。

实例 020　重定向输出流实现程序日志

光盘位置：光盘\MR\02\020

高级

实用指数：★★★☆

实例说明

System 类中的 out 成员变量是 Java 的标准输出流，程序常用它来输出调试信息，out 成员变量被定义为 final 类型的，无法直接重新复制，但是可以通过 setOut()方法来设置新的输出流。本实例利用该方法实现了输出流的重定向，把它指向一个文件输出流，从而实现了日志功能。程序运行后绘制控制台提示运行结束信息，如图 2.3 所示；但是在运行过程中的步骤都保存到了日志文件中，如图 2.4 所示。



图 2.3　控制台运行结果

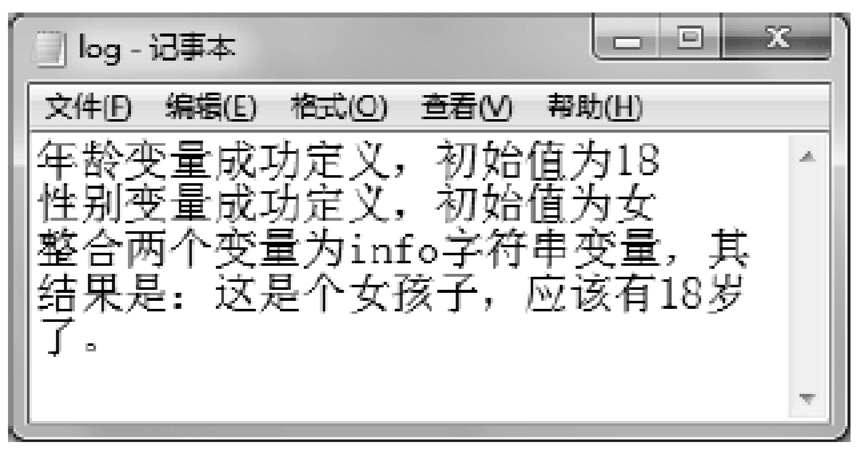


图 2.4　日志文件内容

关键技术

本实例应用的关键技术是调用了 System 类的 setOut()方法改变了输出流，System 类的 out、err 和 in 成员变

量是 final 类型的，不能直接赋值，要通过相应的方法来改变流。下面分别介绍改变这 3 个成员变量的方法。

（1）setOut()方法

该方法用于重新分配 System 类的标准输出流，该方法的声明如下：

```
public static void setOut(PrintStream out)
```

参数说明

out：新的 PrintStream 输出流对象。

（2）setErr()方法

该方法将重新分配 System 类的标准错误输出流，该方法的声明如下：

```
public static void setErr(PrintStream err)
```

参数说明

err：新的 PrintStream 输出流对象。

（3）setIn()方法

该方法将重新设置 System 类的 in 成员变量，即标准输入流。

设计过程

创建 RedirectOutputStream 类，编写该类的 main()主方法，在该方法中保存 System 类的 out 成员变量为临时变量，然后创建一个新的文件输出流，并把这个输出流设置为 System 类新的输出流。在程序关键位置输出调试信息，这些调试信息将通过新的输出流保存到日志文件中。最后恢复原有输出流并输出程序运行结束信息，关键代码如下：

```java
import java.io.FileNotFoundException;
import java.io.PrintStream;
public class RedirectOutputStream {
    public static void main(String[] args) {
        try {
            PrintStream out = System.out;//保存原输出流
            PrintStream ps=new PrintStream("./log.txt");//创建文件输出流
            System.setOut(ps);//设置使用新的输出流
            int age=18;//定义整型变量
            System.out.println("年龄变量成功定义，初始值为 18");
            String sex="女";//定义字符串变量
            System.out.println("性别变量成功定义，初始值为女");
            //整合两个变量
            String info="这是个"+sex+"孩子，应该有"+age+"岁了。";
            System.out.println("整合两个变量为 info 字符串变量，其结果是："+info);
            System.setOut(out);//恢复原有输出流
            System.out.println("程序运行完毕，请查看日志文件。");
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```

秘笈心法

参考本实例的做法，可以把 err 标准错误输出流也重定向到其他位置。例如，可以定义在与标准输出流相同的文件输出流中，但是在输出错误信息时，添加“警告：”字样，这样可以为日志添加信息级别。

实例 021	自动类型转换与强制类型转换 光盘位置：光盘\MR\02\021	高级 实用指数：★★★☆

实例说明

Java 基本数据类型之间存在自动类型转换与强制类型转换两种转换方法。本实例将演示这两种类型转换的

用法，实例运行结果如图 2.5 所示。注意 long 类型向低类型 short 转换时发生的数据丢失。

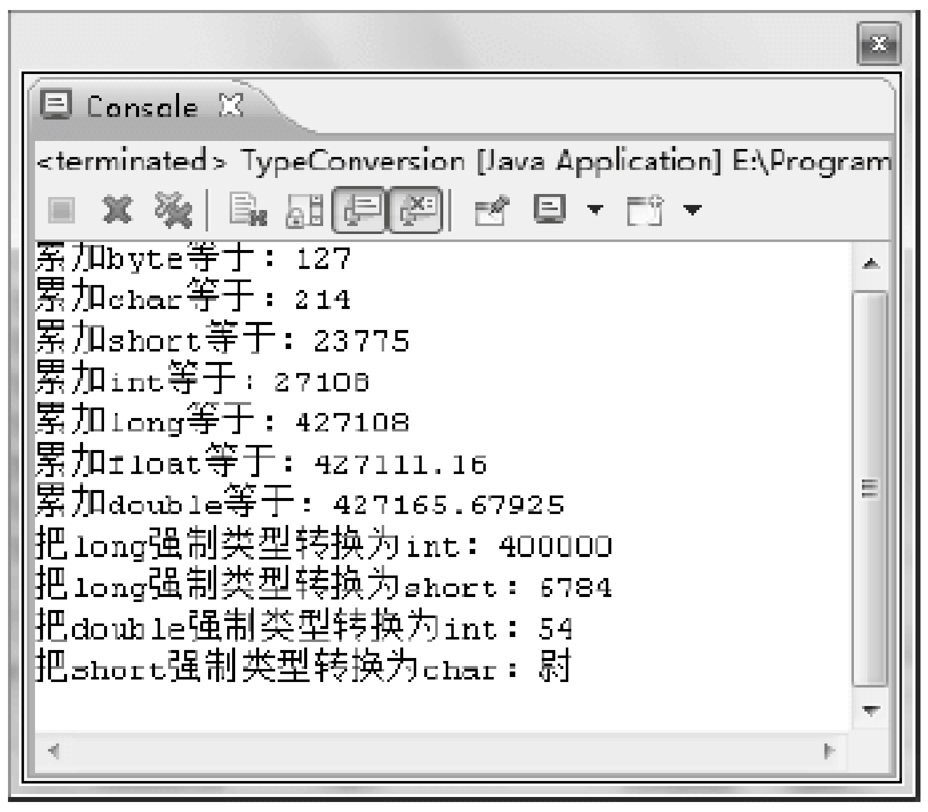


图 2.5　实例运行结果

设计过程

创建 TypeConversion 类，在该类的主方法中创建各种基本类型的变量，在输出语句中分别输出所有变量累加值。注意每次累加值的数据类型，所有整数运算都被自动转换为 int 类型之后再进行运算，所有浮点数值都被自动转换为 double 类型再进行运算。最后把高类型数据向低类型数据进行强制类型转换，并注意运算结果是否丢失数据，关键代码如下：

```
public class TypeConversion {
    public static void main(String[] args) {
        byte b = 127;
        char c = 'W';
        short s = 23561;
        int i = 3333;
        long l = 400000L;
        float f = 3.14159F;
        double d = 54.523;
        //低类型向高类型自动转换
        System.out.println("累加 byte 等于：" + b);
        System.out.println("累加 char 等于：" + (b + c));
        System.out.println("累加 short 等于：" + (b + c + s));
        System.out.println("累加 int 等于：" + (b + c + s + i));
        System.out.println("累加 long 等于：" + (b + c + s + i + l));
        System.out.println("累加 float 等于：" + (b + c + s + i + l + f));
        System.out.println("累加 double 等于：" + (b + c + s + i + l + f + d));
        System.out.println("把 long 强制类型转换为 int：" + (int) l);          //高类型到低类型的强制转换
        System.out.println("把 int 强制类型转换为 short：" + (short) l);       //高类型到低类型转换会丢失数据
        System.out.println("把 double 强制类型转换为 int：" + (int) d);        //实数到整数转换将舍弃小数部分
        System.out.println("把 short 强制类型转换为 char：" + (char) s);       //整数到字符类型的转换将获取对应编码的字符
    }
}
```

秘笈心法

在输出语句中，经常对输出的数字添加一个描述前缀，如“他的年龄是：45”。但是如果 45 是一个数学加法的公式，那么很容易出现错误的运算。首先第一个数字与字符串会通过+符号实现字符串连接，而其后的所有数字加法运算都会被看作字符串的连接操作。解决办法是把所有数字加法用括号括起来。

2.2　运　算　符

实例 022	加密可以这样简单（位运算） 光盘位置：光盘\MR\02\022	高级 实用指数：★★★☆

实例说明

本实例通过位运算的异或运算符“^”把字符串与一个指定的值进行异或运算，从而改变字符串每个字符的值，这样就可以得到一个加密后的字符串，如图 2.6 所示。当把加密后的字符串作为程序输入内容，异或运算会把加密后的字符串还原为原有字符串的值，如图 2.7 所示。

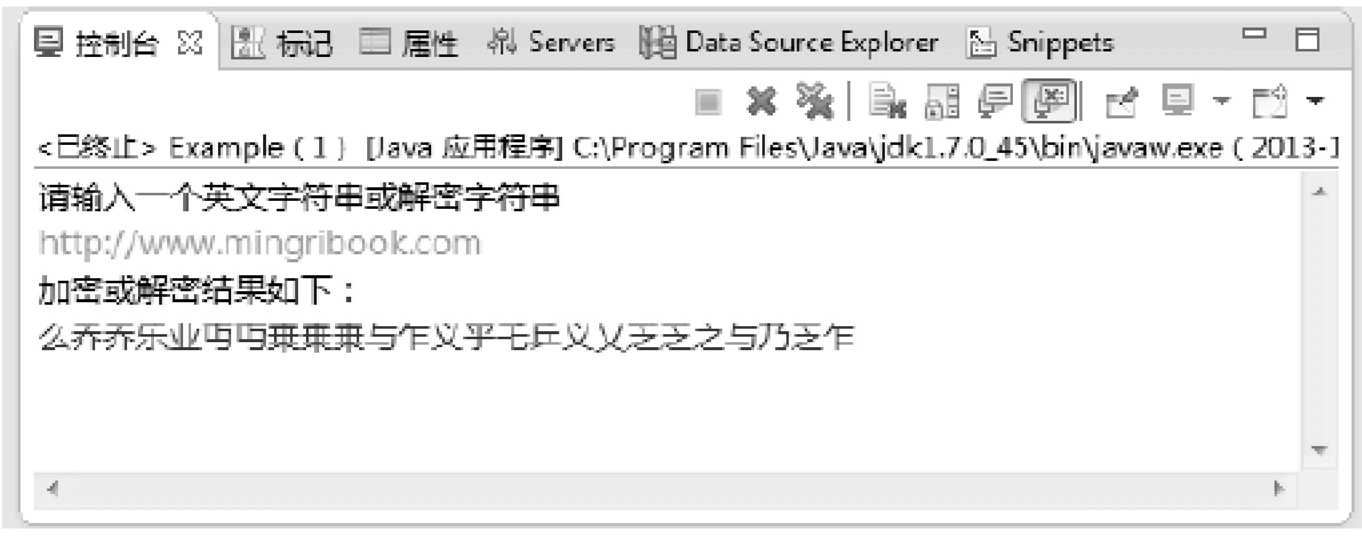


图 2.6　加密效果

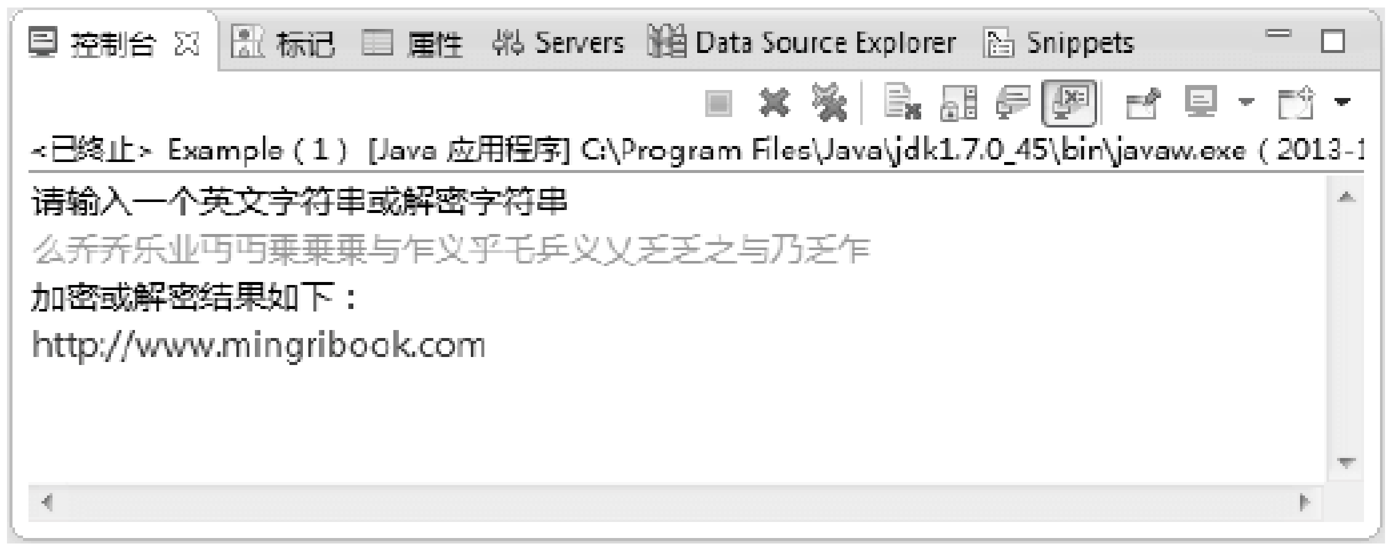


图 2.7　解密效果

设计过程

创建 Example 类，在该类的主方法中创建 System 类的标准输入流的扫描器对象，提示用户输入一个英文的字符串或者要解密的字符串，然后通过扫描器获取用户输入的字符串，经过加密或解密后，把字符串通过错误流输出到控制台，关键代码如下：

```
import java.util.Scanner;
public class Example {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("请输入一个英文字符串或解密字符串");
        String password = scan.nextLine();            //获取用户输入
        char[] array = password.toCharArray();        //获取字符数组
        for (int i = 0; i < array.length; i++) {      //遍历字符数组
            array[i] = (char) (array[i] ^ 20000);     //对每个数组元素进行异或运算
        }
        System.out.println("加密或解密结果如下：");
        System.err.println(new String(array));        //输出密钥
    }
}
```

说明： 程序最后使用标准错误输出流不是用于输出错误信息，而是利用了其在 Eclipse 控制台以红色显示的特性来突出显示。

秘笈心法

灵活运用位运算可以实现很多高级、高效的算法。例如，一个数字的位移运算，每左移 n 位就等于这个数乘以 2 的 n 次方，每右移 n 位就等于这个数除以 2 的 n 次方。而且这个算法非常快。

实例 023	用三元运算符判断奇数和偶数 光盘位置：光盘\MR\02\023	高级 实用指数：★★★☆

实例说明

三元运算符是 if…else 条件语句的简写格式，它可以完成简单的条件判断。本实例利用这个三元运算符实现了奇偶数的判断，程序要求用户输入一个整数，然后程序判断是奇数还是偶数并输出到控制台中，实例运行结果如图 2.8 所示。

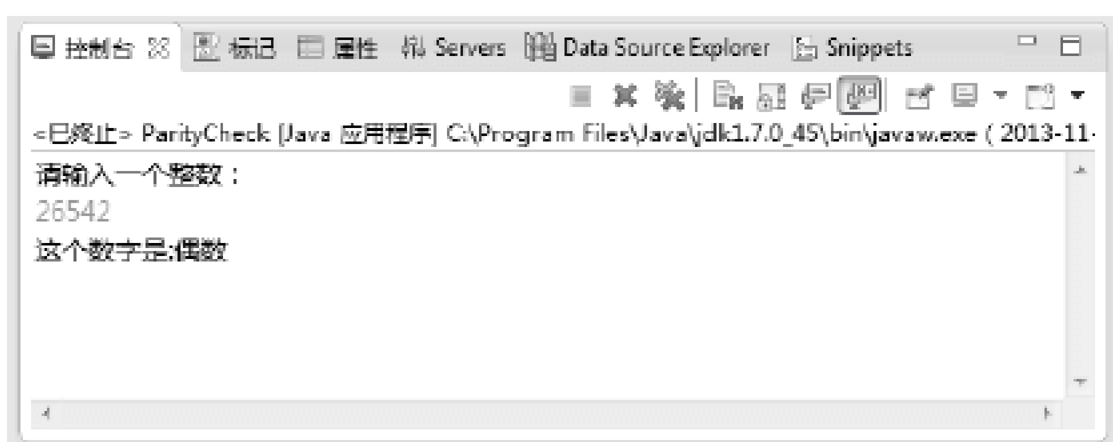


图 2.8　实例运行结果

关键技术

本实例的关键内容就是以三元运算符实现简单的条件判断，其语法格式如下：

```
条件运算?运算结果 1:运算结果 2;
```

如果条件运算结果为 true，返回值就是运算结果 1，否则返回结果 2。

另外，本实例使用扫描器的 nextLong()方法直接获取整型数据，避免了类型转换等业务代码，该方法的声明如下：

```
public long nextLong()
```

该方法返回一个 long 类型的数值，这个数是从扫描器封装的输入流中获取的。

设计过程

创建 ParityCheck 类，在该类的主方法中创建标准输入流的扫描器对象，提示用户输入一个整数，并通过扫描器的方法来接收一个整数，通过三元运算符判断该数字与 2 的余数，如果余数为 0 说明其是偶数，否则是奇数，关键代码如下：

```
import java.util.Scanner;
public class ParityCheck {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);      //创建输入流扫描器
        System.out.println("请输入一个整数：");
        long number = scan.nextLong();              //获取用户输入的整数
        String check = (number % 2 == 0) ? "这个数字是:偶数" : "这个数字是:奇数";
        System.out.println(check);
    }
}
```

秘笈心法

%运算符的用途非常广泛，它能够实现数据分页，最简单的方法是可以通过计算奇偶数的方法把数组交叉分成两个数组。它还可以限制数字的范围，如(*N*%5==0)可以限制数字 *N* 在 0～4 的范围内。

实例 024 更精确地使用浮点数

光盘位置：光盘\MR\02\024

高级

实用指数：★★★☆

实例说明

浮点运算的典型实例是货币运算，在商品金额计算中，经常会涉及小数运算。例如，某个商品的价格是 1.10 元，而顾客现有金额是 2 元整。在计算机中所有数字都是使用二进制进行存储的，而二进制无法精确地表示所有的小数，所以使用基本数据类型进行小数运算会有一些误差，本实例将通过 BigDecimal 类实现精确的小数运算，实例运行结果如图 2.9 所示。

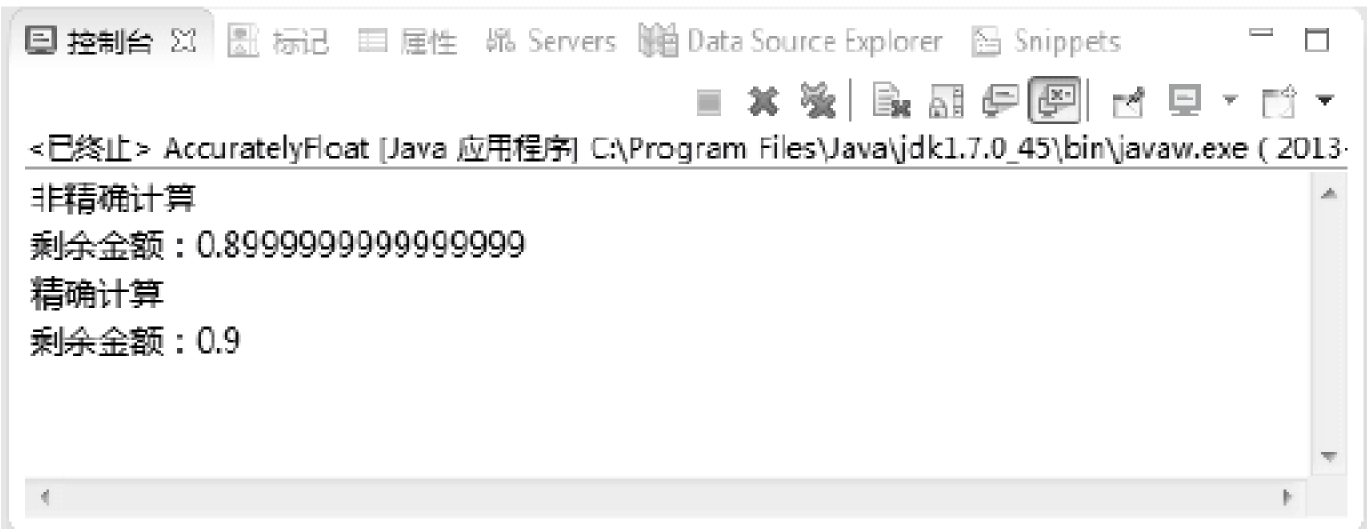


图 2.9　实例运行结果

关键技术

本实例在完成浮点数精确计算的过程中使用了 BigDecimal 类，它用于大数字的精确计算。本实例调用了该类的 subtract()方法实现减法运算。下面介绍该类的运算方法。

（1）加法

该方法实现两个 BigDecimal 类实例对象的加法运算，并将运算结果作为方法的返回值。该方法的声明如下：

```
public BigDecimal add(BigDecimal augend)
```

参数说明

augend：与当前对象执行加法的操作数。

（2）减法

该方法实现两个 BigDecimal 类实例对象的减法运算，并将运算结果作为方法的返回值。该方法的声明如下：

```
public BigDecimal subtract(BigDecimal subtrahend)
```

参数说明

subtrahend：被当前对象执行减法的操作数。

（3）乘法

该方法实现两个 BigDecimal 类实例对象的乘法运算，并将运算结果作为方法的返回值。该方法的声明如下：

```
public BigDecimal multiply(BigDecimal multiplicand)
```

参数说明

multiplicand：乘法运算中的乘数。

（4）除法

该方法实现两个 BigDecimal 类实例对象的除法运算，并将运算结果作为方法的返回值。该方法的声明如下：

```
public BigDecimal divide(BigDecimal divisor)
```

参数说明

divisor：除法运算中的除数。

设计过程

创建 AccuratelyFloat 类，在该类的主方法中创建 double 类型的浮点数变量并输出它们相减的运算结果，然后以 BigDecimal 类的实例再一次完成同样的运算，对比运行结果哪个更精确，关键代码如下：

```
import java.math.BigDecimal;
public class AccuratelyFloat {
    public static void main(String[] args) {
        double money = 2;                                   //现有金额
        double price = 1.1;                                 //商品价格
        double result=money - price;
        System.out.println("非精确计算");
        System.out.println("剩余金额："+result);            //输出运算结果
        //精确浮点数的解决方法
        BigDecimal money1 = new BigDecimal("2");            //现有金额
        BigDecimal price1 = new BigDecimal("1.1");          //单击商品
        BigDecimal result1=money1.subtract(price1);
        System.out.println("精确计算");
        System.out.println("剩余金额："+result1);           //输出精确结果
    }
}
```

技巧：这里创建 BigDecimal 类的实例时，在构造方法中一定要使用数字字符串作为参数；如果直接使用浮点数或该类型的变量作为参数，那么构造方法接收的是经过二进制存储的浮点数，那样就会是不精确的浮点数。

秘笈心法

对于商业程序的开发，一定要注意其中的货币运算。因为计算机无法通过二进制精确地表示所有小数，所以计算机中的小数运算会有一定的误差。虽然误差非常小，但是货币运算可能会操作多个有误差的运算结果，长期的数据累计会造成更大的误差，特别是银行使用的系统不允许任何微小的误差，所以读者应熟练掌握 BigDecimal 类的用法。

实例 025	不用乘法运算符实现 2×16 光盘位置：光盘\MR\02\025	高级 实用指数：★★★☆

实例说明

程序开发中常用的乘法运算是通过“*”运算符或者 BigDecimal 类的 multiply()方法实现的。但是本实例将介绍在这两种方法之外如何实现乘法，而且实现的运算效率非常高，实例运行结果如图 2.10 所示。

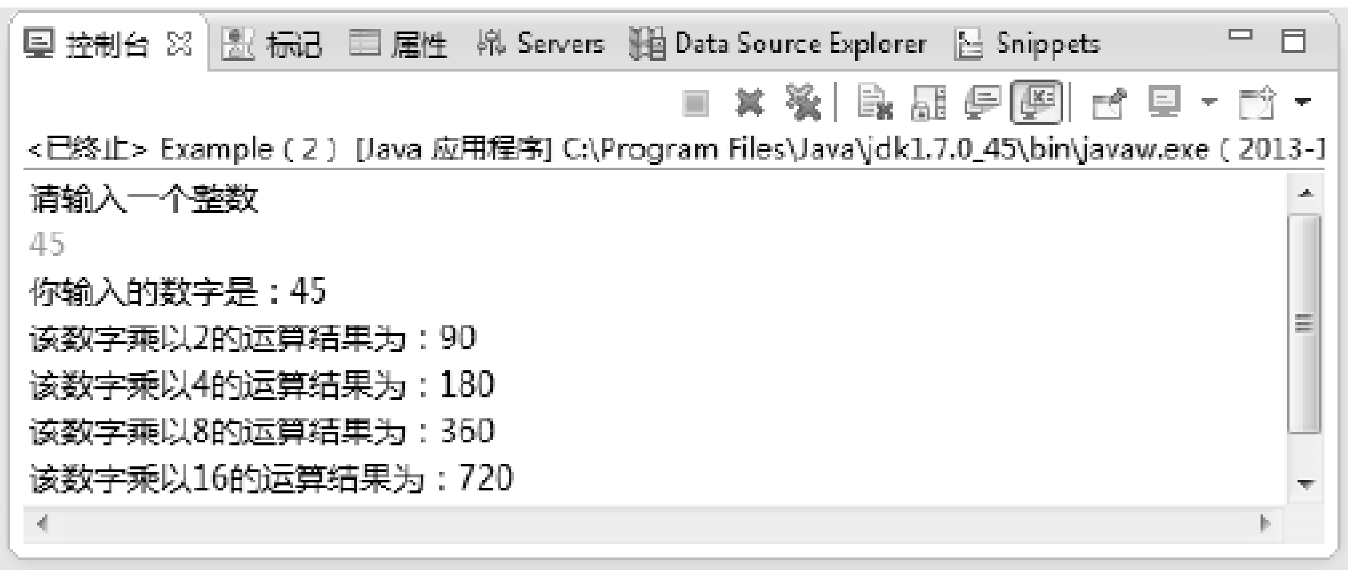


图 2.10　实例运行结果

设计过程

创建 Example 类，在该类的主方法中接收用户输入的一个整数，然后对该整数执行位运算中的左移操作，并输出运算结果，关键代码如下：

```
import java.util.Scanner;
public class Example {
    public static void main(String[] args) {
        Scanner scan=new Scanner(System.in);//创建扫描器
        System.out.println("请输入一个整数");
        long number = scan.nextLong();//获取输入的整数
        System.out.println("你输入的数字是："+number);
        System.out.println("该数字乘以 2 的运算结果为："+(number<<1));
        System.out.println("该数字乘以 4 的运算结果为："+(number<<2));
        System.out.println("该数字乘以 8 的运算结果为："+(number<<3));
        System.out.println("该数字乘以 16 的运算结果为："+(number<<4));
    }
}
```

秘笈心法

通过本实例可以看出，一个整数每次执行位移运算中的左移运算 n 次，相当于这个整数乘以 2 的 n 次方。相反，如果执行右移 n 次运算，则相当于这个整数除以 2 的 n 次方。

实例 026　实现两个变量的互换（不借助第 3 个变量）

光盘位置：光盘\MR\02\026

高级

实用指数：★★★☆

实例说明

变量的互换常见于数组排序算法中，当判断两个数组元素需要交互时，将创建一个临时变量来共同完成互换，临时变量的创建增加了系统资源的消耗。如果需要交换的是两个整数类型的变量，那么可以使用更高效的方法，本实例演示了如何省略临时变量（第 3 个变量）实现两个整数类型变量的高效互换，实例运行结果如图 2.11 所示。

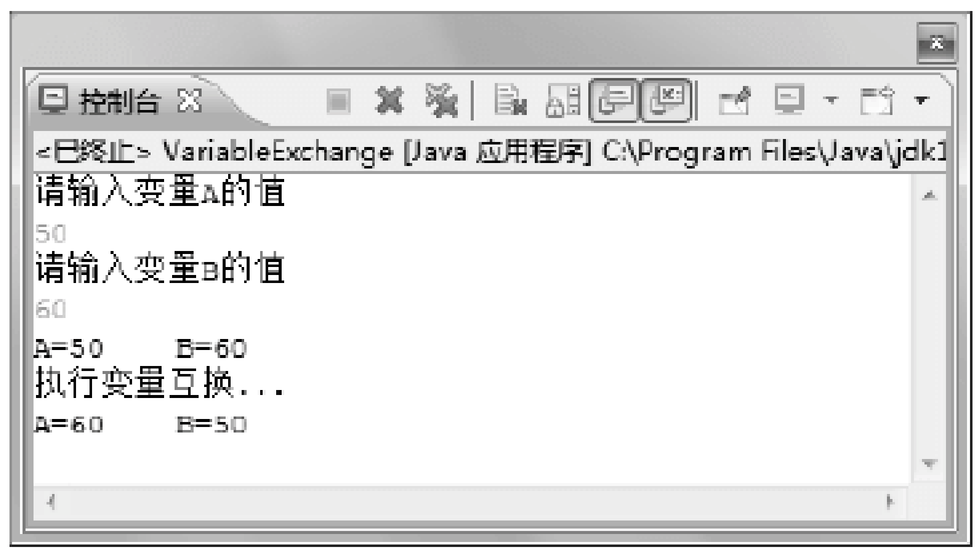


图 2.11　实例运行结果

设计过程

创建 VariableExchange 类，在该类的主方法中创建扫描器对象接收用户输入两个变量值，然后通过位运算中的异或运算符“^”实现两个变量的互换，关键代码如下：

```
import java.util.Scanner;
public class VariableExchange {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);      //创建扫描器
        System.out.println("请输入变量 A 的值");
        long A = scan.nextLong();                   //接收第一个变量值
        System.out.println("请输入变量 B 的值");
```

```
        long B = scan.nextLong();                     //接收第二个变量值
        System.out.println("A=" + A + "\tB=" + B);
        System.out.println("执行变量互换...");
        A = A ^ B;                                    //执行变量互换
        B = B ^ A;
        A = A ^ B;
        System.out.println("A=" + A + "\tB=" + B);
    }
}
```

秘笈心法

异或^和其他位运算符并不会改变变量本身的值，即“A^B;”没有任何意义，必须将其运算结果赋值给一个变量。

2.3 条件语句

实例 027 判断某一年是否为闰年

光盘位置：光盘\MR\02\027

高级

实用指数：★★★☆

实例说明

为了弥补因人为历法规定造成的年度天数与地球实际公转周期的时间差，设立了 366 天的闰年，闰年的二月份有 29 天。本实例通过程序计算用户输入的年份是否为闰年，实例运行结果如图 2.12 所示。

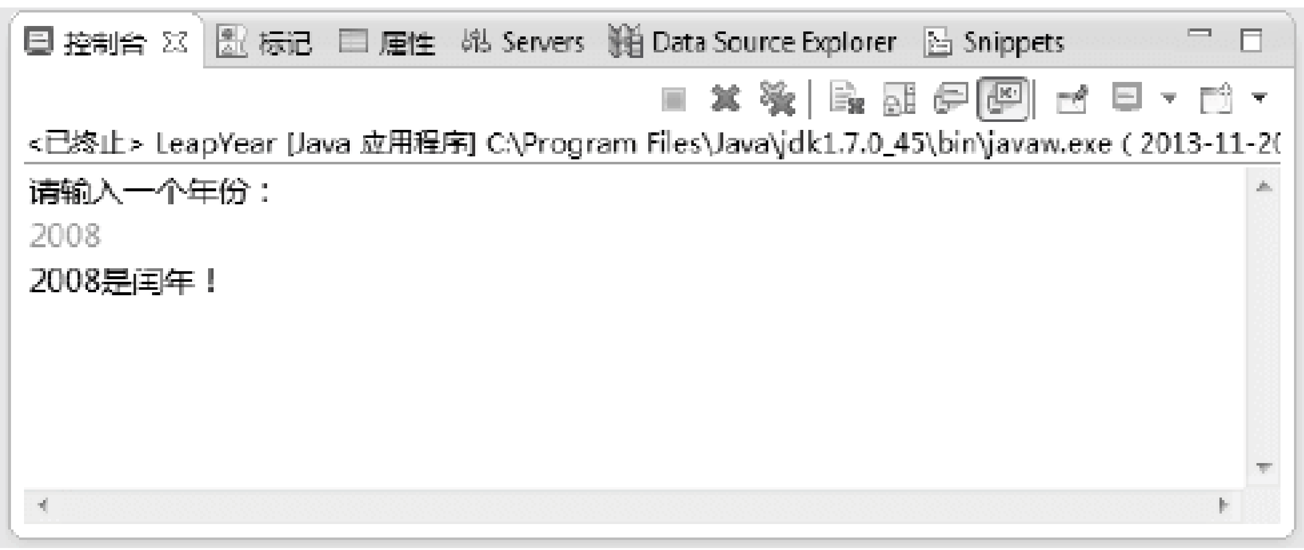


图 2.12 实例运行结果

关键技术

本实例计算闰年的关键技术是其公式。满足两种条件的整数可以称为闰年，第一，能被 4 整除但不能被 100 整除；第二，能被 400 整除。

该公式用 Java 语法实现的格式如下：

```
year % 4 == 0 && year % 100 != 0 || year % 400 == 0
```

设计过程

创建 LeapYear 类，在该类的主方法中接收用户输入的一个整数年份，然后通过闰年计算公式，判断这个年份是否为闰年，并在控制台输出判断结果，关键代码如下：

```
import java.util.Scanner;
public class LeapYear {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("请输入一个年份：");
        long year = scan.nextLong();//接收用户输入
        if (year % 4 == 0 && year % 100 != 0 || year % 400 == 0) { //是闰年
            System.out.print(year + "是闰年！");
        } else { //不是闰年
            System.out.print(year + "不是闰年！");
        }
    }
}
```

秘笈心法

三元运算符（?:）是 if…else 语法的一个简洁写法，可以根据需求来决定使用哪种。前者常用于赋值判断，后者常用于业务流程。

实例 028 验证登录信息的合法性

光盘位置：光盘\MR\02\028

高级

实用指数：★★★☆

实例说明

大多系统登录模块都会接收用户通过键盘输入的登录信息，这些登录信息将会被登录模块验证，如果使用的是指定的用户名与密码，则允许程序登录；否则将用户拒之门外。本实例通过 if…else 进行多条件判断实现了登录信息验证，实例运行结果如图 2.13 所示。

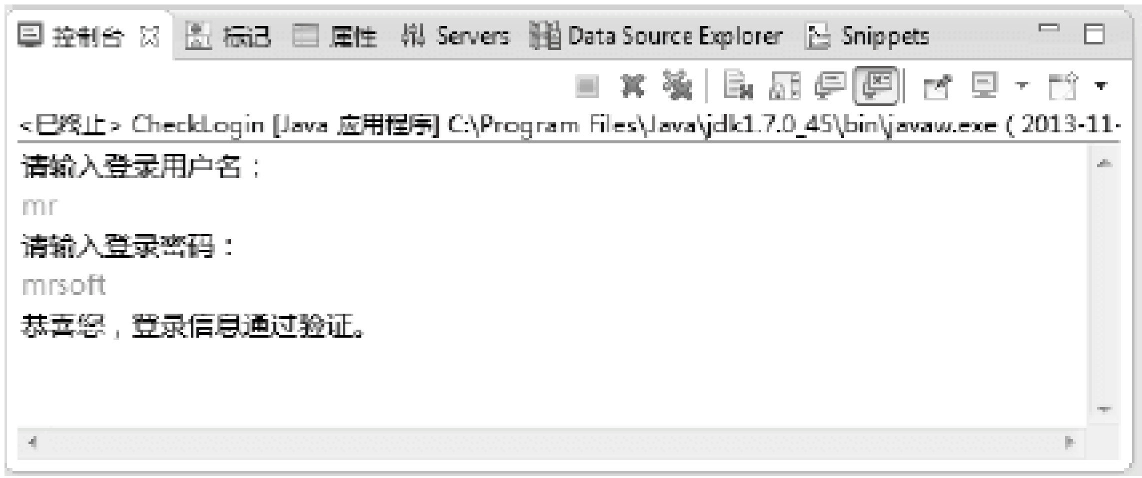


图 2.13　输入合法登录信息的效果

设计过程

创建 CheckLogin 类，在该类的主方法中接收用户输入的登录用户名与密码，然后通过 if 条件语句分别判断用户名与密码，并输出登录验证结果，关键代码如下：

```
import java.util.Scanner;
public class CheckLogin {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);          //创建扫描器
        System.out.println("请输入登录用户名：");
        String username = scan.nextLine();              //接收用户输入登录名
        System.out.println("请输入登录密码：");
        String password = scan.nextLine();              //接收用户输入登录密码
        if (!username.equals("mr")) {                   //判断用户名的合法性
            System.out.println("用户名非法。");
        } else if (!password.equals("mrsoft")) {        //判断密码的合法性
            System.out.println("登录密码错误。");
        } else {                                        //通过以上两个条件判断则默认通过登录验证
            System.out.println("恭喜您，登录信息通过验证。");
        }
    }
}
```

秘笈心法

字符串属于对象而非基本数据类型，不能够使用==来判断两个字符串是否相等，所以它需要通过 equals() 方法来判断两个字符串内容是否相同，正如本实例对用户名和密码的判断那样。如果使用==判断的将是两个字符串对象的内存地址，而非字符串内容。

实例 029 为新员工分配部门

光盘位置：光盘\MR\02\029

高级

实用指数：★★★☆

实例说明

本实例根据用户输入的信息确定员工应该分配到哪个部门。实例中需要根据用户输入进行多条件判断，所以采用了 switch 语句，实例运行结果如图 2.14 所示。

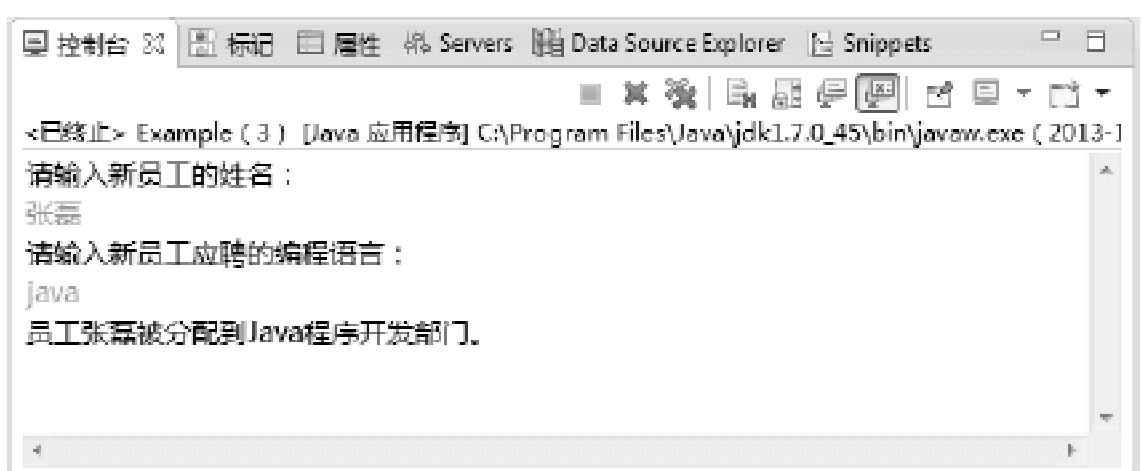


图 2.14　实例运行结果

关键技术

本实例的关键技术在于 switch 多分支语句的使用，该语句只支持对常量的判断，而常量又只能是 Java 的基本数据类型，虽然

在以后的 JDK 版本中可以对 String 类的字符串对象进行判断，但是就目前项目的需求也有对字符串进行多条件判断的。本实例采取的做法是对字符串的哈希码进行判断，也就是把 String 类的 hashCode()方法返回值作为 switch 语法的表达式，case 关键字之后跟随的是各种字符串常量的哈希码整数值。

设计过程

创建 Example 类，在该类的主方法中创建标准输入流的扫描器，通过扫描器获取人事部门输入的姓名与应聘编程语言，然后根据每个语言对应的哈希码来判断分配部门，关键代码如下：

```
import java.util.Scanner;
public class Example {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("请输入新员工的姓名：");
        String name = scan.nextLine();          //接收员工名称
        System.out.println("请输入新员工应聘的编程语言：");
        String language = scan.nextLine();      //接收员工应聘的编程语言
        //根据编程语言确定员工分配的部门
        switch (language.hashCode()) {
            case 3254818:       //Java 的哈希码
            case 2301506:       //Java 的哈希码
            case 2269730:       //Java 的哈希码
                System.out.println("员工"+name+"被分配到 Java 程序开发部门。");
                break;
            case 3104:          //c#的哈希码
            case 2112:          //C#的哈希码
                System.out.println("员工"+name+"被分配到 C#项目维护组。");
                break;
            case -709190099:    //asp.net 的哈希码
            case 955463181:     //Asp.net 的哈希码
            case 9745901:       //ASP.NET 的哈希码
                System.out.println("员工"+name+"被分配到 Asp.net 程序测试部门。");
                break;
            default:
                System.out.println("本公司不需要" + language + "语言的程序开发人员。");
        }
    }
}
```

秘笈心法

在 switch 语法中每个 case 关键字可以作为一个条件分支，但是对于多个条件采取相同业务处理的情况，可以把多个 case 分支关联在一起，省略它们之间的 break 语句，而在最后一个相同的 case 分支中实现业务处理并执行 break 语句，就像本实例中应用的那样。

实例 030	用 switch 语句根据消费金额计算折扣 光盘位置：光盘\MR\02\030	高级 实用指数：★★★☆

实例说明

编写程序，应用 switch 语句计算累计消费金额达到一定数额时，享受不同的折扣价格，实例运行结果如图 2.15 所示。

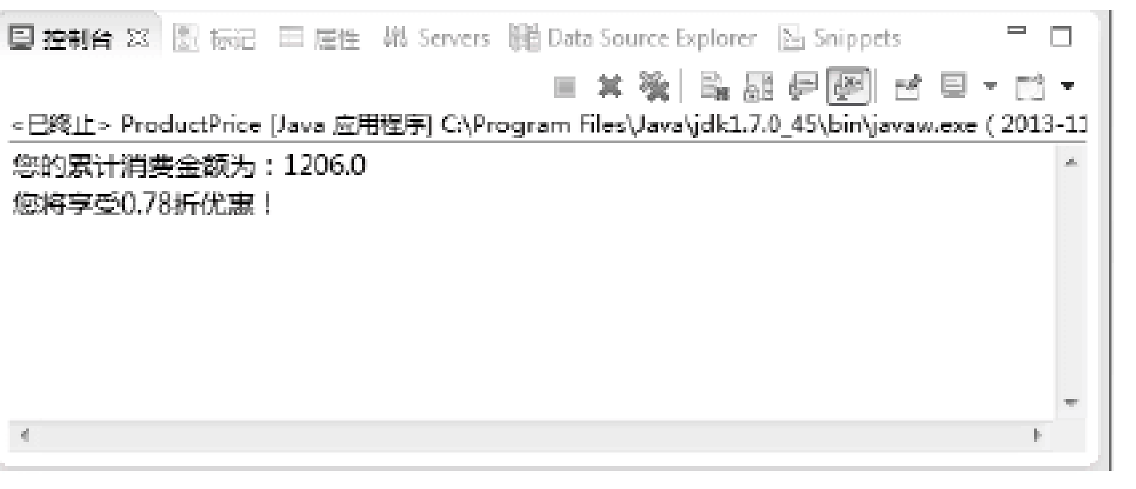


图 2.15　实例运行结果

设计过程

创建 ProductPrice 类，在该类的主方法中实现本实例的业务代码。该方法首先假设一个用户消费总额的变量 money，并初始化一个折扣变量 rebate，然后经过运算来获得用户等级，对不同的等

级给予不同的折扣优惠，程序关键代码如下：

```
public class ProductPrice {
    public static void main(String[] args) {
        float money = 1206;                                    //金额
        float rebate = 0f;                                     //折扣
        if (money > 200) {
            int grade = (int) money / 200;                     //等级
            switch (grade) {                                   //根据等级计算折扣比例
                case 1:
                    rebate = 0.95f;
                    break;
                case 2:
                    rebate = 0.90f;
                    break;
                case 3:
                    rebate = 0.85f;
                    break;
                case 4:
                    rebate = 0.83f;
                    break;
                case 5:
                    rebate = 0.80f;
                    break;
                case 6:
                    rebate = 0.78f;
                    break;
                case 7:
                    rebate = 0.75f;
                    break;
                case 8:
                    rebate = 0.73f;
                    break;
                case 9:
                    rebate = 0.70f;
                    break;
                case 10:
                    rebate = 0.65f;
                    break;
                default:
                    rebate = 0.60f;
            }
        }
        System.out.println("您的累计消费金额为：" + money);    //输出消费金额
        System.out.println("您将享受" + rebate + "折优惠！");   //输出折扣比例
    }
}
```

秘笈心法

在程序开发中经常使用的都是正数，负数因为使用得少，常常被忽略。例如，“*N*%2==1”本来是用来计算数字 *N* 是否为奇数的，但是开发者由于没有考虑到负数的情况，导致这个算法的失败，因为任何负数应用这个算法都会等于-1。

实例 031	判断用户输入月份的季节 光盘位置：光盘\MR\02\031	高级 实用指数：★★★☆

实例说明

本实例根据用户输入的月份来判断季节，这是一个最典型的实践 switch 语法的例子。通过这个例子可以完全掌握 switch 语法的用法与技巧，实例运行结果如图 2.16 所示。

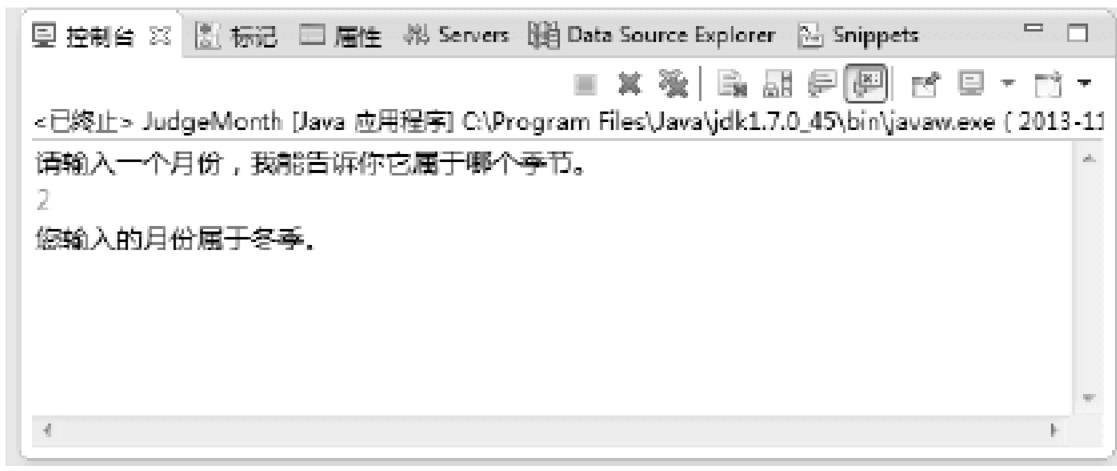


图 2.16　实例运行结果

设计过程

创建 JudgeMonth 类，在该类的主方法中创建扫描器接收用户输入的月份数字，然后判断该月份属于哪个季节并输出到控制台，对于非法月份也要给出提示，程序关键代码如下：

```
import java.util.Scanner;
public class JudgeMonth {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);              //创建扫描器
        //提示用户输入月份
        System.out.println("请输入一个月份，我能告诉你它属于哪个季节。");
        int month = scan.nextInt();                         //接收用户输入
        switch (month) {                                    //判断月份属于哪个季节
            case 12:
            case 1:
            case 2:
                System.out.print("您输入的月份属于冬季。");
                break;
            case 3:
            case 4:
            case 5:
                System.out.print("您输入的月份属于春季");
                break;
            case 6:
            case 7:
            case 8:
                System.out.print("您输入的月份属于夏季");
                break;
            case 9:
            case 10:
            case 11:
                System.out.print("您输入的月份属于秋季");
                break;
            default:
                System.out.print("你那有" + month + "月份吗？");
        }
    }
}
```

秘笈心法

switch 语句的每个 case 关键字都用于判断一个常量并做出相应的业务处理，熟练掌握 switch 语句之后可以组合多个 case 来完成多条件的处理，就是多个常量结果执行相同的业务处理，就像本实例中的那样。

2.4　循 环 控 制

实例 032	使用 while 与自增运算符循环遍历数组 光盘位置：光盘\MR\02\032	高级 实用指数：★★★☆

实例说明

本实例利用自增运算符结合 while 循环获取每个数组元素的值，然后把它们输出到控制台中。其中自增运

算符控制索引变量的递增，实例运行结果如图 2.17 所示。

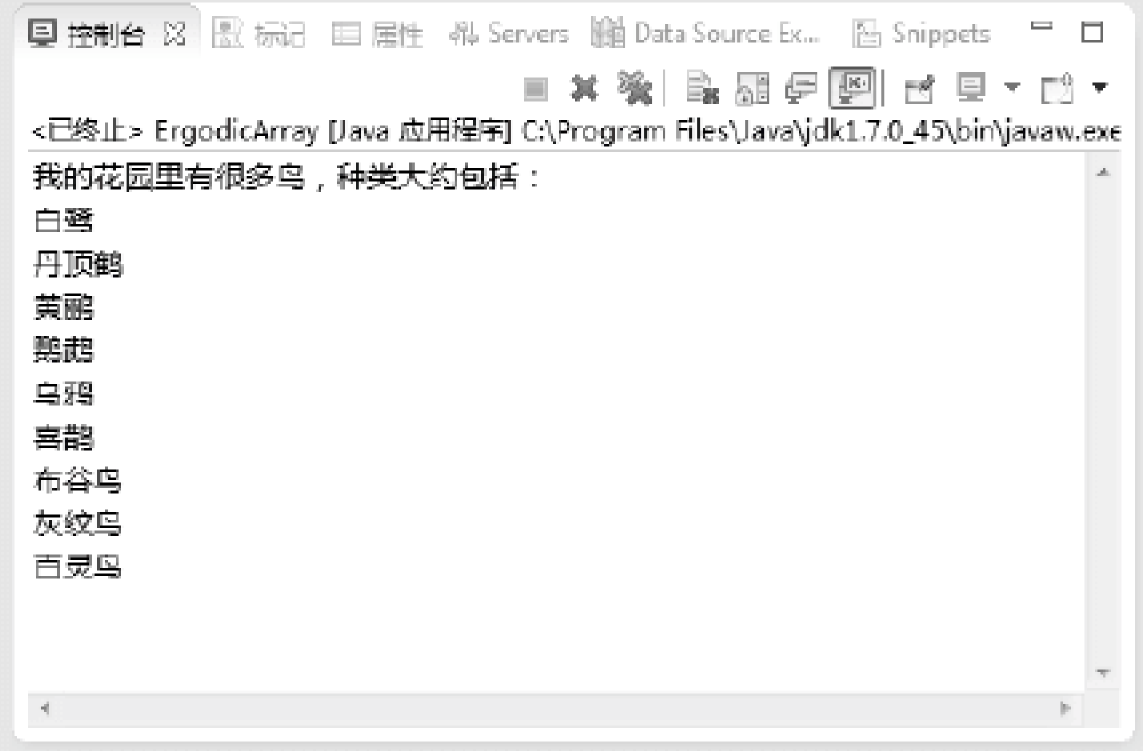


图 2.17　实例运行结果

设计过程

创建 ErgodicArray 类，在该类的主方法中创建一个鸟类数组，然后创建一个索引变量，这个变量用于指定数组下标，随着该索引的递增，while 循环会逐步获取每个数组的元素并输出到控制台中，关键代码如下：

```
public class ErgodicArray {
    public static void main(String[] args) {
        //创建鸟类数组
        String[] aves = new String[] { "白鹭", "丹顶鹤", "黄鹂", "鹦鹉", "乌鸦", "喜鹊",
                "布谷鸟", "灰纹鸟", "百灵鸟" };
        int index = 0;//创建索引变量
        System.out.println("我的花园里有很多鸟，种类大约包括：");
        while (index < aves.length) {//遍历数组
            System.out.println(aves[index++]);//自增索引值
        }
    }
}
```

秘笈心法

自增自减运算符分前置与后置两种，其中前置运算如“++index”会先将 index 的值递增，然后再使用递增后的值；而后置运算如“index++”会首先使用该变量的值，然后再把变量值递增。

实例 033	使用 for 循环输出杨辉三角 光盘位置：光盘\MR\02\033	高级 实用指数：★★★☆

实例说明

杨辉三角由数字排列，可以把它看作一个数字表，其基本特性是两侧数值均为 1，其他位置的数值是其正上方的数值与左上角数值之和。本实例通过数组来实现这个杨辉三角，其运行结果如图 2.18 所示。

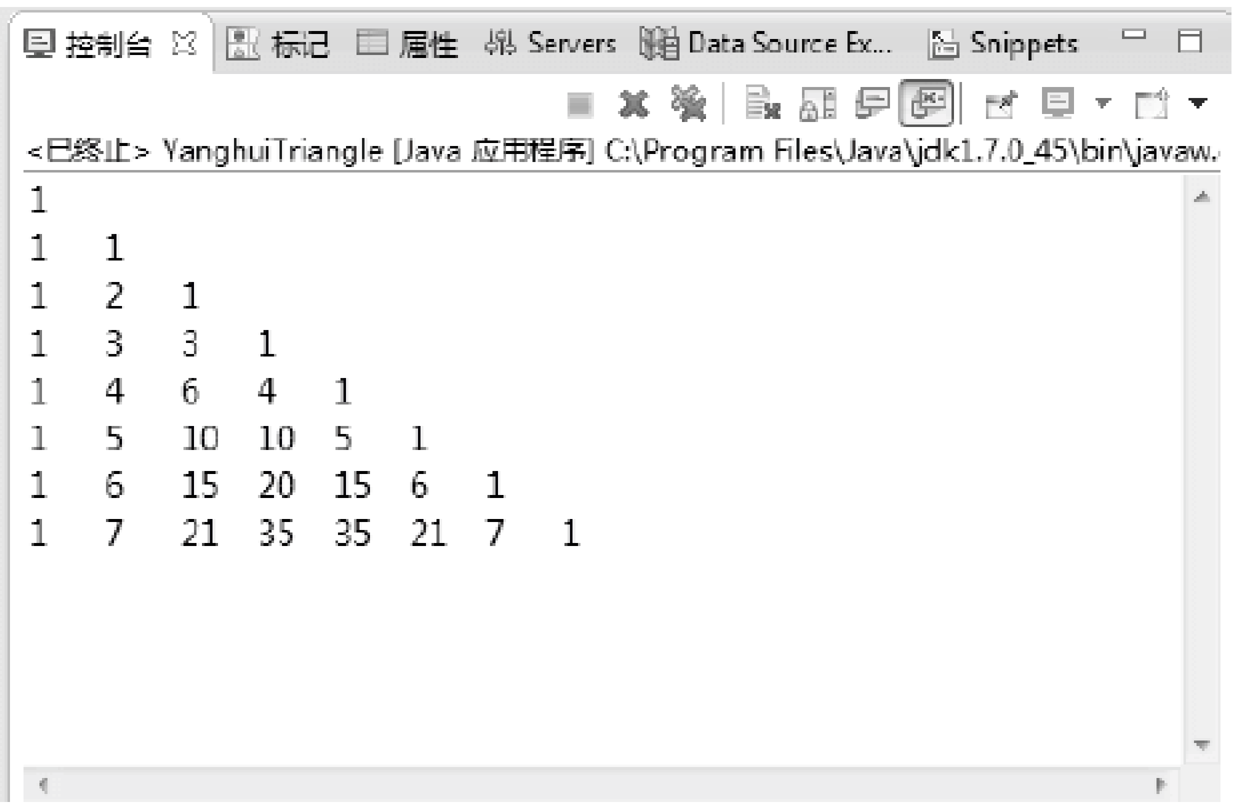


图 2.18　杨辉三角

设计过程

创建 YanghuiTriangle 类，在该类的主方法中创建一个二维数组，并指定二维数组的第一维长度，这个数组用于存放杨辉三角的数值表，再通过双层 for 循环来实现第二维数组的长度，然后计算整个数组的每个元素的值，程序关键代码如下：

```
public class YanghuiTriangle {
    public static void main(String[] args) {
        int triangle[][]=new int[8][];//创建二维数组
        //遍历二维数组的第一层
        for (int i = 0; i < triangle.length; i++) {
            triangle[i]=new int[i+1];//初始化第二层数组的大小
            //遍历第二层数组
            for(int j=0;j<=triangle[i].length-1;j++){
                //将两侧的数组元素赋值为 1
                if(i==0||j==0||j==triangle[i].length-1){
                    triangle[i][j]=1;
                }else{//其他数值通过公式计算
                    triangle[i][j]=triangle[i-1][j]+triangle[i-1][j-1];
                }
                //输出数组元素
                System.out.print(triangle[i][j]+"\t");
            }
            System.out.println();
        }
    }
}
```

秘笈心法

Java 语言中的二维数组其实是一维数组的每个元素都是另一个一维数组，所以第二维数组的长度可以任意，就像本实例中那样。这比其他语言的数组更灵活，而且多维数组也是如此。

实例 034　使用嵌套循环在控制台上输出九九乘法表

光盘位置：光盘\MR\02\034

高级

实用指数：★★★☆

实例说明

Java 基本语法中的 for 循环非常灵活并且可以嵌套使用，其中双层 for 循环是程序开发中使用最频繁的，常用于操作表格数据。对于行数与列数相同的表格操作代码比较简单，但是类似九九乘法表就不好控制了，因为它的列数要与行数对应，可以说这个表格是个三角形。本实例通过双层循环输出了这个九九乘法表，效果如图 2.19 所示。（在面试与等级考试中常出现这类题目）

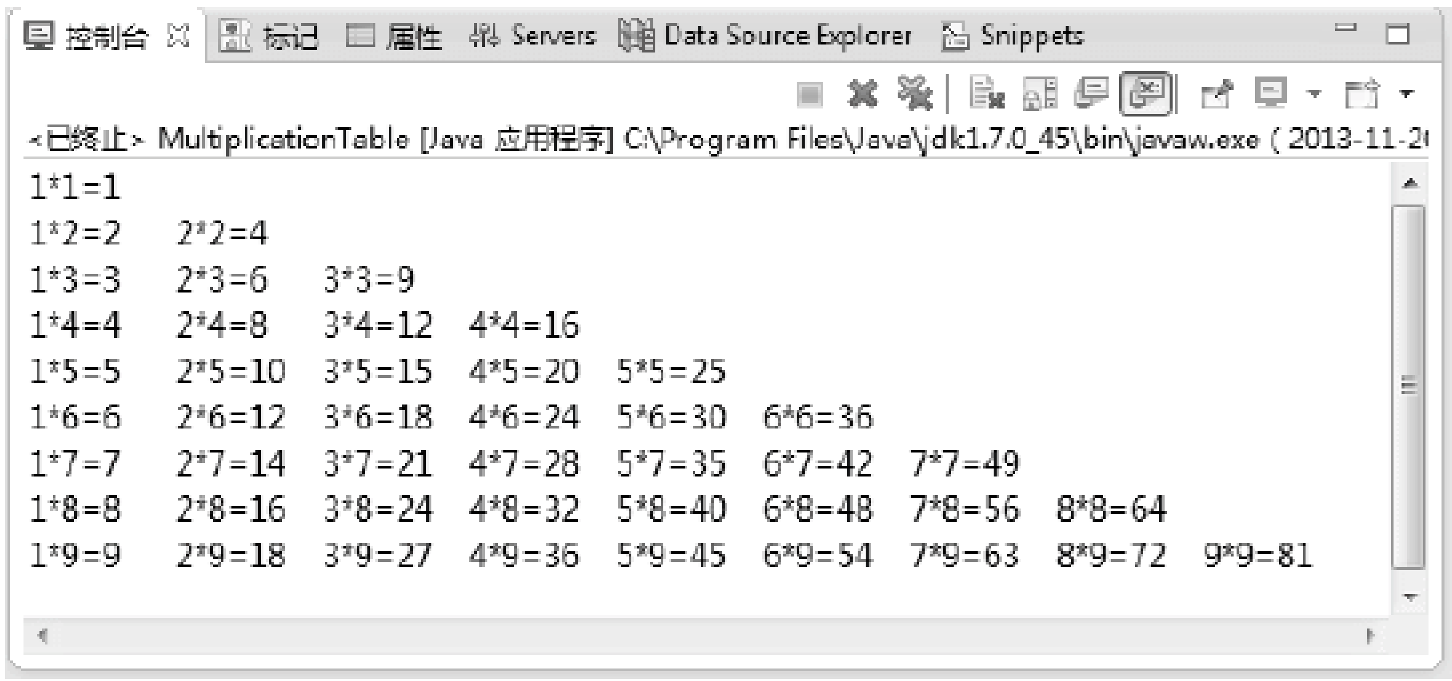


图 2.19　九九乘法表

设计过程

创建 MultiplicationTable 类，在该类的主方法中创建双层 for 循环，第一层 for 循环也称为外层循环，它用于控制表格的行；第二层循环也称为内层循环，它用于控制表格的列。这里第二层循环的控制变量非常重要，它的条件判断是列数要等于行数的最大值，然后输出内层与外层循环控制变量的乘积，这样就实现了九九乘法表，程序关键代码如下：

```
public class MultiplicationTable {
    public static void main(String[] args) {
```

```
        for(int i=1;i<=9;i++){//循环控制变量从 1 遍历到 9
            for(int j=1;j<=i;j++){//第二层循环控制变量与第一层最大索引相等
                //输出计算结果但不换行
                System.out.print(j+"*"+i+"="+i*j+"\t");
            }
            System.out.println();//在外层循环中换行
        }
    }
}
```

秘笈心法

循环语句可以完成复杂的运算，也可以用于控制程序的递归流程，而多层循环可以实现更加复杂的业务逻辑，是学习编程必须掌握的一种应用。在处理有规则的大量数据时，应该考虑使用多层循环来优化程序代码，但是建议添加详细的代码注释，便于以后的维护与修改工作。

实例 035 用 while 循环计算 1+1/2!+1/3!…1/20!

光盘位置：光盘\MR\02\035

高级

实用指数：★★★☆

实例说明

本实例在计算阶乘的算法之上应用 while 循环语句计算 1+1/2!+1/3!…1/20!的和。如果使用基本数据类型 double 是无法精确地显示运算结果的，所以本实例使用了 BigDecimal 类的实例来完成这个运算，实例运行结果如图 2.20 所示。

说明：由于本实例的运行结果精度非常高，小数位数过长，所以设置了特殊的控制台执行，读者的运行结果可能是单行的数字。

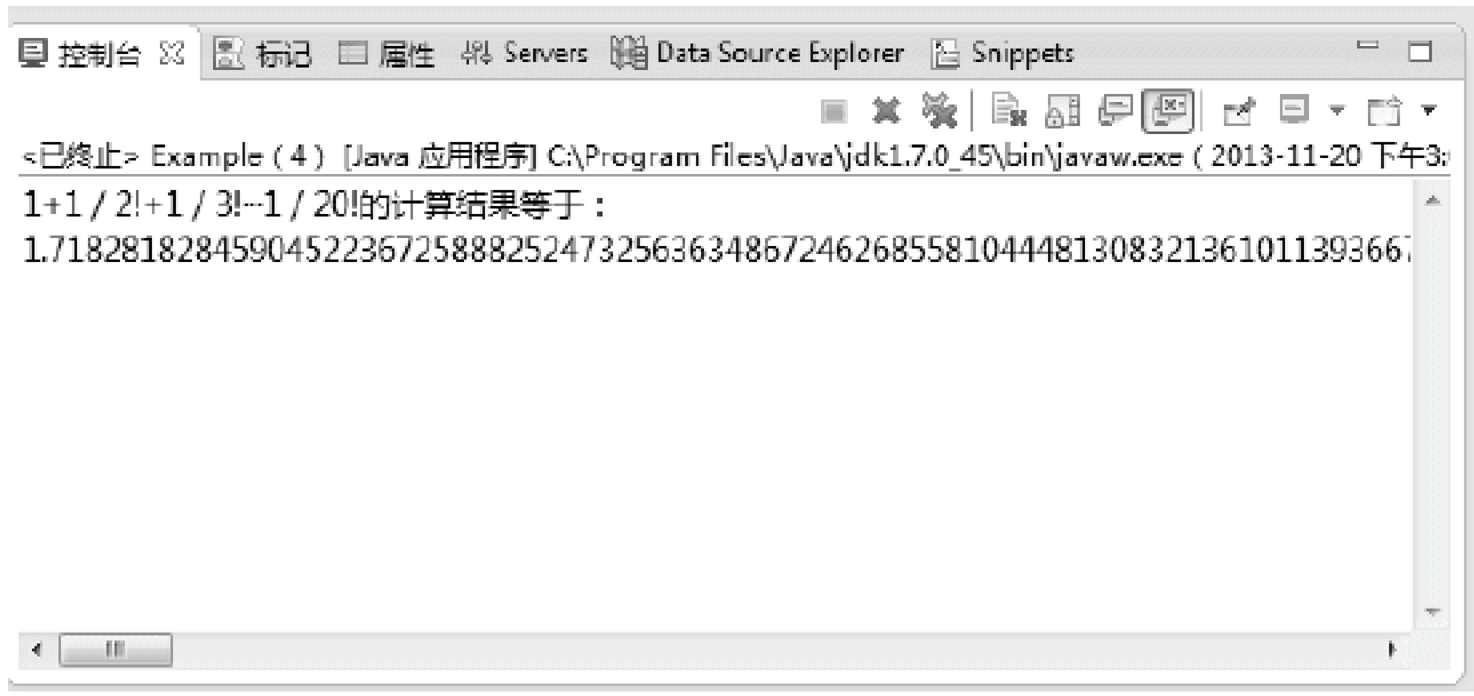


图 2.20　实例运行结果

设计过程

创建 Example 类，在该类的主方法中创建保存总和的 sum 变量和计算阶乘的 factorial 变量，为保证计算结果的精度，这两个变量都是 BigDecimal 类的实例对象。然后通过 while 实现 20 次循环，并完成计算，程序代码如下：

```
import java.math.BigDecimal;
public class Example {
    public static void main(String args[]) {
        BigDecimal sum = new BigDecimal(0.0);                          //和
        BigDecimal factorial = new BigDecimal(1.0);                    //阶乘项的计算结果
        int i = 1;                                                     //循环增量
        while (i <= 20) {
            sum = sum.add(factorial);                                  //累加各项阶乘的和
            ++i;                                                       //i 加 1
            factorial = factorial.multiply(new BigDecimal(1.0 / i));   //计算阶乘项
        }
       System.out.println("1+1 / 2!+1 / 3!…1 / 20!的计算结果等于：\n" + sum);  //输出计算结果
    }
}
```

秘笈心法

对于高精度要求或者运算数较大的计算，应该使用 BigDecimal 类实现，否则 Java 基本类型的数据无法保证浮点数的精度，也无法对超出其表示范围的数字进行运算。

实例 036　用 for 循环输出空心的菱形

光盘位置：光盘\MR\02\036

高级

实用指数：★★★☆

实例说明

本实例在输出菱形的基础上加大难度，输出空心的菱形图案，在等级考试与公司面试时出现过类似题目。本实例的目的在于熟练掌握 for 循环的嵌套使用，实例运行结果如图 2.21 所示。

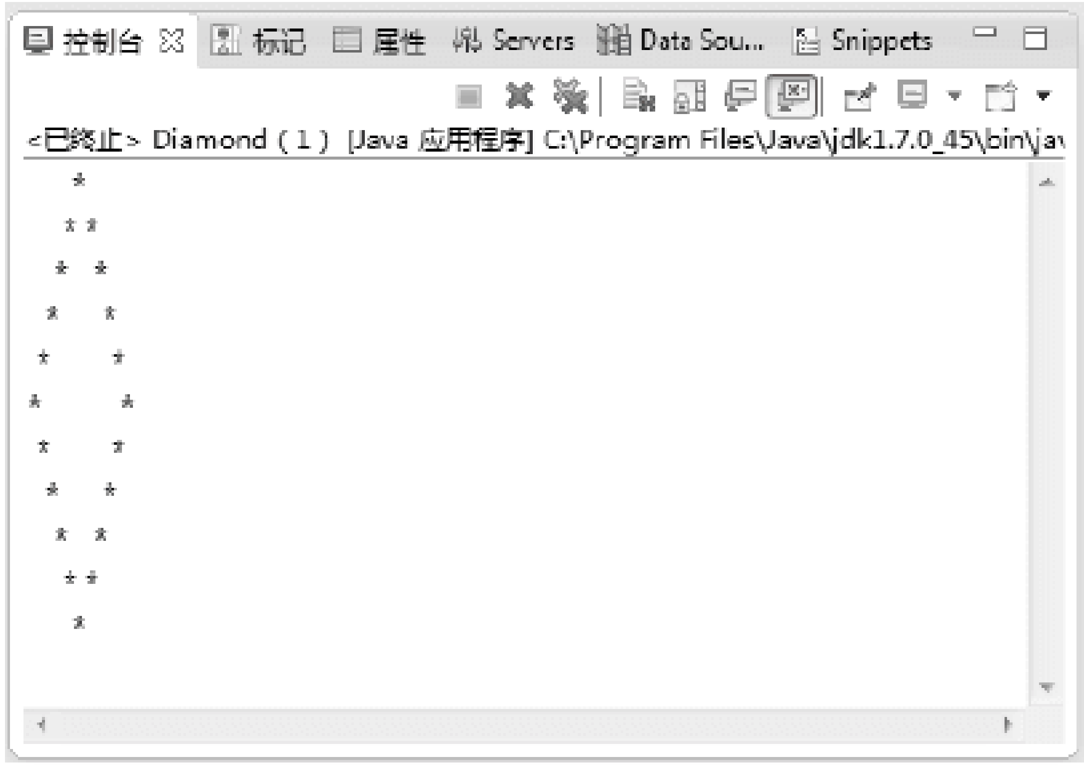


图 2.21　实例运行结果

设计过程

创建 Diamond 类，在该类的主方法中调用 printHollowRhombus()方法完成 10 行的空心菱形输出，其中 printHollowRhombus()方法是实例中自定义的，该方法使用两个双层 for 循环分别输出菱形的上半部分与下半部分，程序关键代码如下：

```
public class Diamond {
    public static void main(String[] args) {
        printHollowRhombus(10);
    }
    public static void printHollowRhombus(int size) {
        if (size % 2 == 0) {
            size++;//计算菱形大小
        }
        for (int i = 0; i < size / 2 + 1; i++) {
            for (int j = size / 2 + 1; j > i + 1; j--) {
                System.out.print(" ");//输出左上角位置的空白
            }
            for (int j = 0; j < 2 * i + 1; j++) {
                if (j == 0 || j == 2 * i) {
                    System.out.print("*");//输出菱形上半部边缘
                } else {
                    System.out.print(" ");//输出菱形上半部空心
                }
            }
            System.out.println("");
        }
        for (int i = size / 2 + 1; i < size; i++) {
            for (int j = 0; j < i - size / 2; j++) {
                System.out.print(" ");//输出菱形左下角空白
            }
            for (int j = 0; j < 2 * size - 1 - 2 * i; j++) {
                if (j == 0 || j == 2 * (size - i - 1)) {
                    System.out.print("*");//输出菱形下半部边缘
                } else {
                    System.out.print(" ");//输出菱形下半部空心
                }
            }
            System.out.println("");
        }
    }
}
```

秘笈心法

for 循环中有 3 个表达式，这 3 个表达式都是可选的，也就是说 for 循环可以没有表达式。例如，for(;;)这样的 for 循环将是一个无限循环，读者在使用 for 循环时应注意避免无限循环。

实例 037 foreach 循环优于 for 循环

光盘位置：光盘\MR\02\037

高级

实用指数：★★★☆

实例说明

JDK 1.5 为 Java 添加了新的 for 循环 foreach。它是原有 for 循环遍历数据的一种简写形式，使用的关键字依然是 for，但是参数格式不同。本实例使用 foreach 循环分别遍历集合对象与数组，并把元素输出到控制台，实例运行结果如图 2.22 所示。

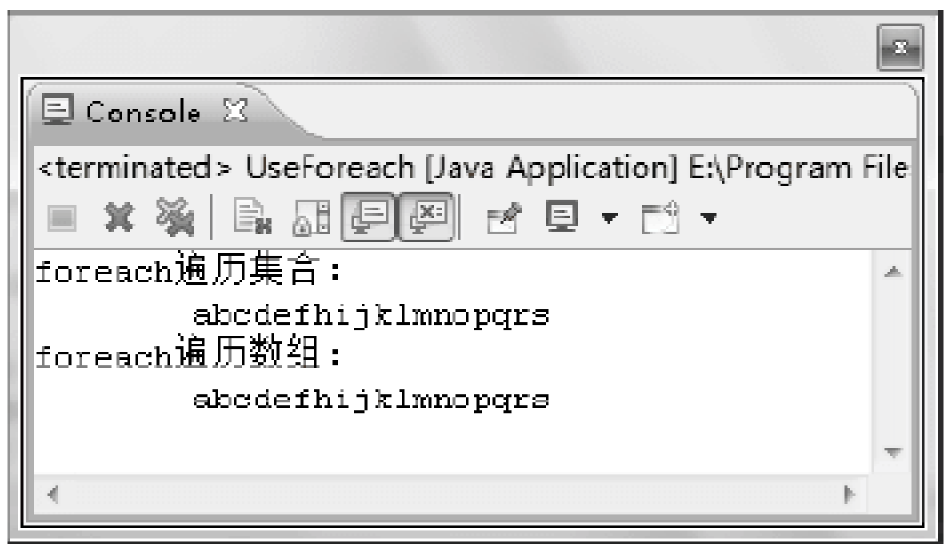


图 2.22　实例运行结果

关键技术

foreach 循环是 for 循环的一种简写格式，只用于遍历数据集合或数组，其语法格式如下：

```
for ( Type e : collections ) {
    //对变量 e 的使用
}
```

参数说明

❶ e：其类型 Type 是集合或数组中元素值的类型，该参数是集合或数组 collections 中的一个元素。

❷ collections：要遍历的集合或数组，也可以是迭代器。

说明： 在循环体中使用参数 e，该参数是 foreach 从集合或数组以及迭代器中取得的元素值，元素值是从头到尾进行遍历的。

设计过程

创建 UseForeach 类，在该类的主方法中创建 List 集合对象，并为该对象添加内容，然后使用 foreach 循环遍历该集合输出所有内容，再从 List 集合中提取一个字符串数组，最后使用 foreach 循环遍历该数组，并将所有数组元素输出到控制台，程序关键代码如下：

```
import java.util.ArrayList;
import java.util.List;
public class UseForeach {
    public static void main(String[] args) {
        List<String> list=new ArrayList<String>();//创建 list 集合
        list.add("abc");//初始化 list 集合
        list.add("def");
        list.add("hij");
        list.add("klm");
        list.add("nop");
        list.add("qrs");
        System.out.print("foreach 遍历集合：\n\t");
        for (String string : list) {//遍历 list 集合
            System.out.print(string);//输出集合的元素值
        }
        System.out.println();
        String[] strs=new String[list.size()];
        list.toArray(strs);//创建数组
        System.out.print("foreach 遍历数组：\n\t");
        for (String string : strs) {//遍历数组
            System.out.print(string);//输出数组元素值
        }
    }
}
```

秘笈心法

在 JDK 1.5 之前使用 for 循环对集合、数值和迭代器进行遍历，这需要创建索引变量、条件表达式，这些会

使代码造成混乱，并增加出错的几率，并且每次循环中索引变量或迭代器都会出现 3 次，有两次出错的机会。而且会有一些性能损失，其性能稍微落后于 foreach 循环。所以对于数据集合的遍历，建议使用 foreach 循环完成。

实例 038　终止循环体　　高级

光盘位置：光盘\MR\02\038　　实用指数：★★★☆

实例说明

循环用于复杂的业务处理可以提高程序的性能和代码的可读性，但是循环中也有特殊情况，如由于某些原因需要立刻中断循环去执行下面的业务逻辑，本实例运行结果如图 2.23 所示。

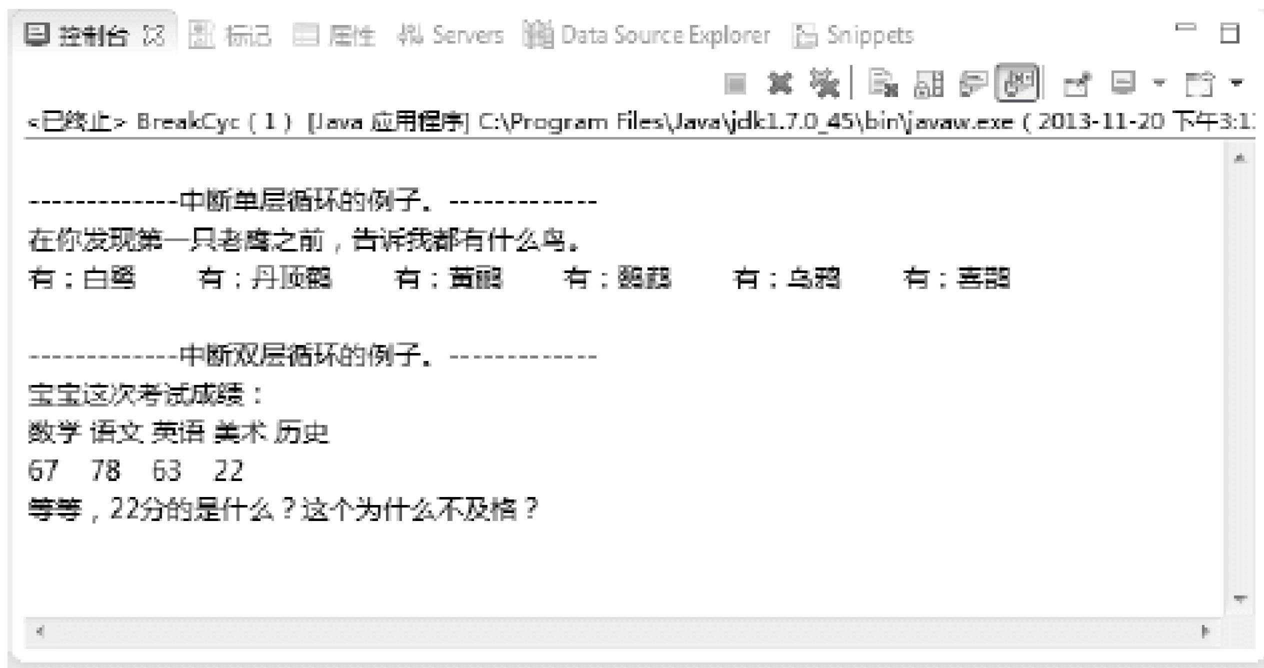


图 2.23　实例运行结果

设计过程

在 Eclipse 中创建一个 Java 项目，在项目中创建 BreakCyc 类，在该类的主方法中创建一个字符串数组，在使用 foreach 遍历时，判断如果发现数组中包含字符串“老鹰”则立刻中断循环。再创建一个整数类型二维数组，使用双层 foreach 循环遍历，当发现第一个小于 60 的数组元素，则立刻中断整个双层循环，而不是内层循环，程序关键代码如下：

```
public class BreakCyc {
    public static void main(String[] args) {
        System.out.println("\n-------------中断单层循环的例子。-------------");
        //创建数组
        String[] array = new String[] { "白鹭", "丹顶鹤", "黄鹂", "鹦鹉", "乌鸦", "喜鹊",
                "老鹰", "布谷鸟", "老鹰", "灰纹鸟", "老鹰", "百灵鸟" };
        System.out.println("在你发现第一只老鹰之前，告诉我都有什么鸟。");
        for (String string : array) {//foreach 遍历数组
            if (string.equals("老鹰"))//如果遇到老鹰
                break;//中断循环
            System.out.print("有：" + string+"          ");//否则输出数组元素
        }
        System.out.println("\n\n-------------中断双层循环的例子。-------------");
        //创建成绩数组
        int[][] myScores = new int[][] { { 67, 78, 63, 22, 66 },
                { 55, 68, 78, 95, 44 }, { 95, 97, 92, 93, 81 } };
        System.out.println("宝宝这次考试成绩：\n 数学\t 语文\t 英语\t 美术\t 历史");
        No1: for (int[] is : myScores) {//遍历成绩表格
            for (int i : is) {
                System.out.print(i + "\t");//输出成绩
                if (i < 60) {//如果中途遇到不及格的，立刻中断所有输出
                    System.out.println("\n 等等，" + i + "分的是什么？这个为什么不及格？");
                    break No1;
                }
            }
            System.out.println();
        }
    }
}
```

秘笈心法

充分利用循环可以提高程序的开发与执行效率，但是如果不注重循环中的算法很容易导致程序的死循环，那将是程序的死穴。所以在循环体中要对可能出现的特殊情况使用 break 语句中断循环。

实例 039 循环体的过滤器

光盘位置：光盘\MR\02\039

高级

实用指数：★★★☆

实例说明

循环体中可以通过 break 语句中断整个循环，这增加了循环的控制能力，但是对于特殊情况还是不够。例如，某些条件下需要放弃部分循环处理，而不是整个循环体。Java 提供了 continue 语句来实现这一功能，continue 可以放弃本次循环体的剩余代码，不执行它们而开始下一轮的循环。本实例利用 continue 语句实现了循环体过滤器，可以过滤“老鹰”字符串，并做相应的处理，但是放弃 continue 语句之后的所有代码，实例运行结果如图 2.24 所示。

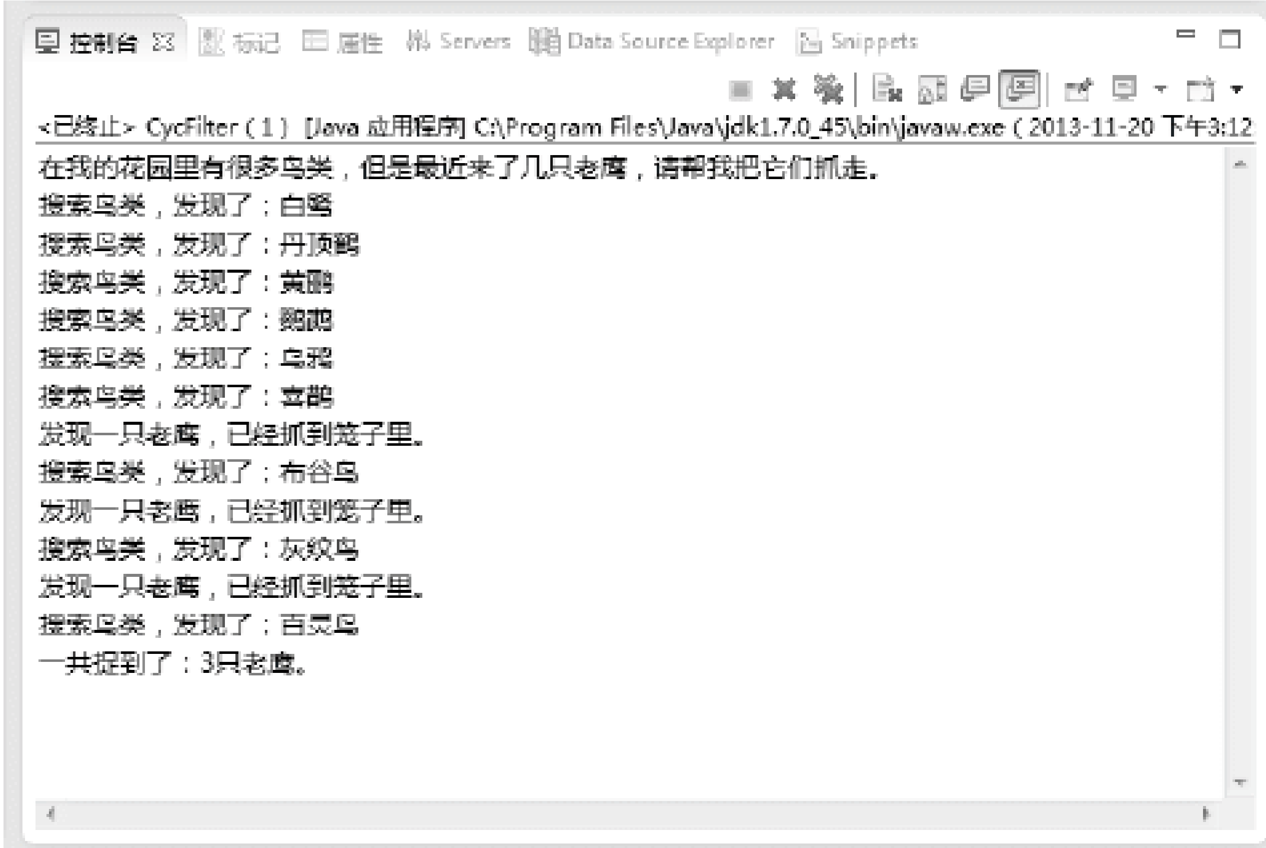


图 2.24　实例运行结果

设计过程

在 Eclipse 项目中创建 CycFilter 类，在该类的主方法中创建鸟类名称的字符串数组，其中包含多个“老鹰”字符串，然后通过 foreach 循环遍历该数组，在循环过程中如果遍历的数组元素是“老鹰”字符串，则输出发现老鹰的信息并过滤循环体之后的所有代码，关键代码如下：

```
public class CycFilter {
    public static void main(String[] args) {
        //创建数组
        String[] array = new String[] { "白鹭", "丹顶鹤", "黄鹂", "鹦鹉", "乌鸦", "喜鹊",
                "老鹰", "布谷鸟", "老鹰", "灰纹鸟", "老鹰", "百灵鸟" };
        System.out.println("在我的花园里有很多鸟类，但是最近来了几只老鹰，请帮我把它们抓走。");
        int eagleCount = 0;
        for (String string : array) {//foreach 遍历数组
            if (string.equals("老鹰")) {//如果遇到老鹰
                System.out.println("发现一只老鹰，已经抓到笼子里。");
                eagleCount++;
                continue;//中断循环
            }
            System.out.println("搜索鸟类，发现了：" + string);//否则输出数组元素
        }
        System.out.println("一共捉到了：" + eagleCount + "只老鹰。");
    }
}
```

秘笈心法

break 语句和 continue 语句都是对循环体的控制语句，它们不仅应用于 for 循环，而且在任何循环体中也可以使用这些语句，灵活使用可以让循环实现更加复杂的运算和业务处理。

实例 040 循环的极限

光盘位置：光盘\MR\02\040

高级

实用指数：★★★☆

实例说明

循环是常用的开发模式，它可以简化业务处理，提高代码编写与程序运行效率，但是要熟练掌握循环中的控制算法，否则容易造成死循环导致程序崩溃。本实例将介绍一个 Java 语言中很难发现的导致程序死循环的实

例，本实例使用 int 整数类型作为循环索引变量，也是循环控制变量，用它来控制循环的次数，测试当这个程序的条件是索引小于等于变量类型的最大值时会发生什么。

■ 设计过程

创建 CycUtmost 类，在该类的主方法中创建 int 整数类型的变量 end，使其等于整数类型的最大值，然后用该值减去 50 开始作为循环的起始点，条件是循环控制变量小于等于 end 变量，在循环体中累加循环计数器，最后循环结束时显示这个计数器，代码如下：

```
public class CycUtmost {
    public static void main(String[] args) {
        int end=Integer.MAX_VALUE;//定义循环终止数
        int start=end-50;//定义循环起始数
        int count=0;//定义循环计数器
        for (int i = start; i <= end; i++) {//执行循环
            count++;//循环计数
        }
        //输出循环计数器
        System.out.println("本次循环次数为："+count);
    }
}
```

✍ **技巧**：读者可能会认为这个程序会循环至少 50 次，然后输出计数器的值，但实际上这个程序的运行结果会导致死循环，因为控制条件是索引小于等于整数类型的最大值，当整数类型达到其最大值再累加 1 时会回到整数类型的最小值，所以它永远不可能大于 end 变量，这样就导致了程序的死循环，所以在程序开发时要注意控制变量的取值范围。

■ 秘笈心法

Java 基本数据类型都有其取值范围，熟悉二进制原理的读者应该能够理解，当超出取值范围时，数值会被截取。例如，本实例中的循环控制变量超出整数类型的最大取值范围时，就会绕回整数类型的最小值。所以在进行条件判断涉及取值边界时，要考虑这个因素。

2.5　常 用 排 序

实例 041	冒泡排序法 光盘位置：光盘\MR\02\041	高级 实用指数：★★★☆

■ 实例说明

冒泡排序是交换排序中最简单的排序方法，其基本思想是：两两比较相邻的关键码，如果反序则交换，直到没有反序的记录为止，本实例运行结果如图 2.25 所示。

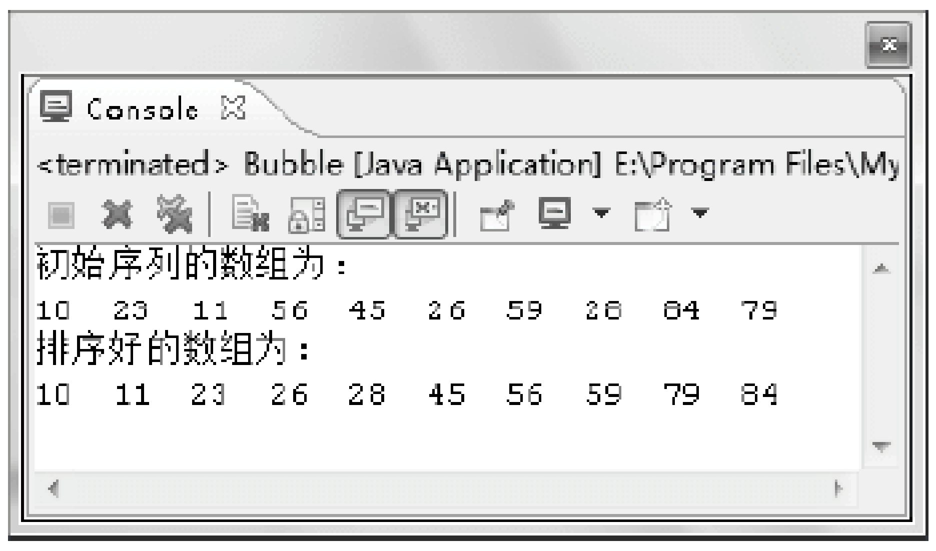


图 2.25　实例运行结果

■ 关键技术

对于冒泡排序，已经知道了其基本思想，下面是对于冒泡排序需要解决的关键问题。第一将整个记录序列分为有序区和无序区。第二对无序区的比较，将使关键码小的记录向前移动，使关键码大的向后移动，一直重复以上操作，直到无序区没有反序的记录。

■ 设计过程

实现冒泡排序的关键代码如下：

```java
package bubble;
public class Bubble {
public static void main(String[] args) {
 int a[]={10,23,11,56,45,26,59,28,84,79};                //给出原始数的序列
 int i,temp;
 System.out.println("初始序列的数组为：");                //输出排序好的数序列
      for(i=0;i<10;i++){
              System.out.print(a[i]+"    ");
      }
      for(i=0;i<9;i++){
         if(a[i]>a[i+1]){                                  //进行两两比较，下面进行符合条件的交换
               temp=a[i];
               a[i]=a[i+1];
               a[i+1]=temp;
         }
      }
      System.out.println("\n"+"排序好的数组为：");     //输出排序好的数序列
      for(i=0;i<10;i++){
              System.out.print(a[i]+"    ");
      }
}
}
```

秘笈心法

冒泡排序实际上就是利用一个中间变量来实现数值的交换。由于这种算法简单易用，因此在实际开发项目过程中，经常用到这种算法。

实例 042　快速排序法

光盘位置：光盘\MR\02\042

高级　实用指数：★★★☆

实例说明

快速排序算法是对冒泡排序算法的一种改进，由于冒泡排序是对相邻的数据进行两两比较，元素的移动次数和比较次数都比较多，而快速排序是从记录序列的两端向中间进行的，所以在元素的移动次数和比较次数上都减少了，本实例运行结果如图 2.26 所示。

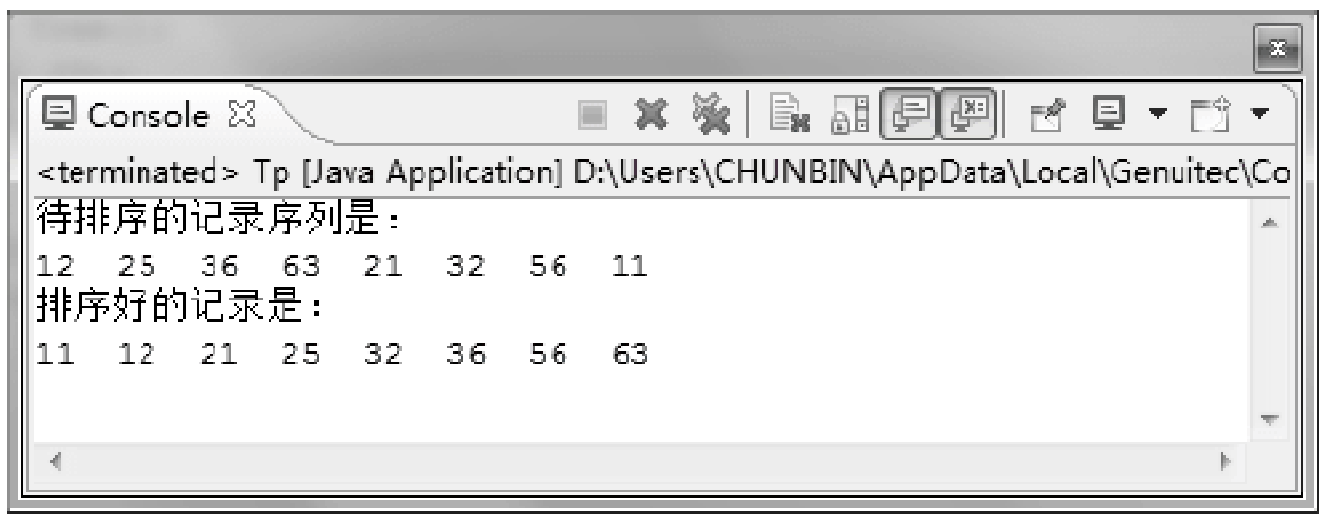


图 2.26　实例运行结果

关键技术

快速排序的基本思想是：首先选定一个轴值（就是比较的基准），将待排序记录分割成独立的两部分，左侧记录的关键码都小于或等于轴值，右侧的记录关键码都大于或等于关键码，然后再对这两部分分别重复上述的过程，直到整个序列有序。

设计过程

实现快速排序一次划分的关键代码如下：

```java
package bubble;
      public class Parti {
      public int partition(int[] r,int first,int end){
            int i,j;
            i=first;j=end;                              //初始化
            while(i<j){
                  while(i<j&&r[i]<=r[j])    j--;        //右侧扫描
                        if(i<j){                         //将较小的记录交换到前面
                              int temp;
                              temp=r[i];
                              r[i]=r[j];
```

```
                        r[j]=temp;
                    }
                while(i<j&&r[i]<r[j]){
                    i++;                          //左侧扫描
                }
                if(i<j){                          //将较大的记录交换到后面
                    int temp;
                    temp=r[i];
                    r[i]=r[j];
                    r[j]=temp;
                }
            }
            return i;//
        }
}
```

秘笈心法

对于快速排序算法，首要解决的就是轴值的确定问题。对此最简单的方法就是用记录的第一个记录作为轴值，但这样不能保证不出现正序或逆序的问题，这样将要对序列进行一次前后调换。所以可以取首值、中值以及末值，选取其中大小居中的作为轴值。

实例 043	选择排序法 光盘位置：光盘\MR\02\043	高级 实用指数：★★★☆

实例说明

选择排序是一类借助“选择”进行排序的方法，其主要思想是：每趟排序在当前待排序序列中选出关键码最小的记录，添加到有序序列中。选择排序比较独特的地方是：记录的移动次数少，本实例运行结果如图 2.27 所示。

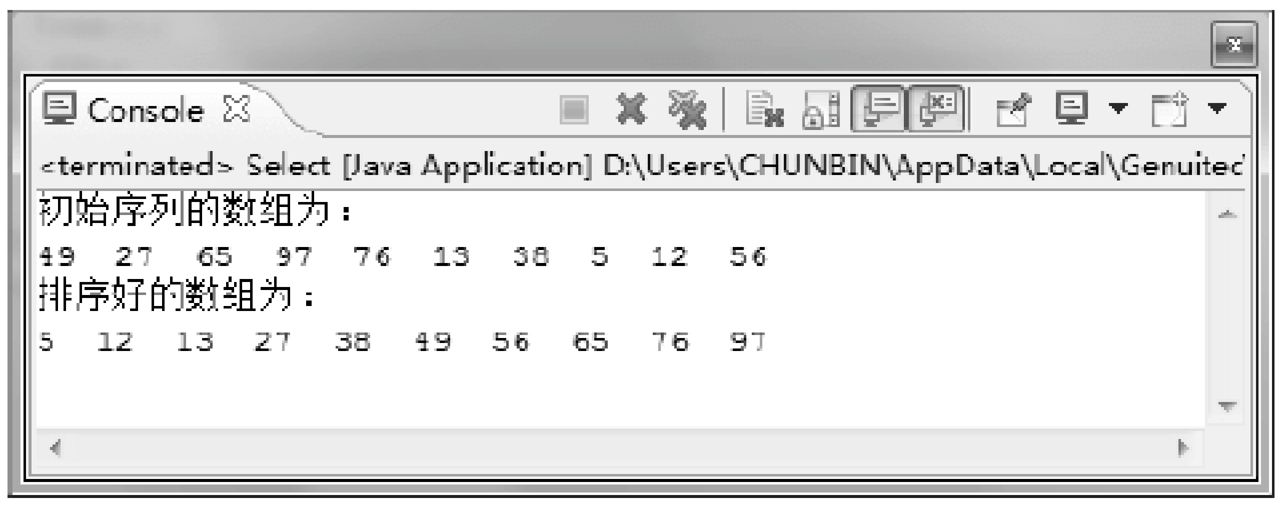


图 2.27　实例运行结果

设计过程

实现选择排序的关键代码如下：

```
package select;
public class Select {
public static void main(String[] args) {
        int r[]={49,27,65,97,76,13,38,5,12,56};
        int i,j,index,temp;
        System.out.println("初始序列的数组为：");
        for(i=0;i<10;i++){                     //对 n 个记录进行 n-1 趟的选择排序
            System.out.print(r[i]+"    ");
    }
        for(i=0;i<9;i++){
            index=i;                           //初始化第 i 趟选择排序的最小记录的指针
            for(j=i+1;j<10;j++){               //在无序区选取最小记录
                if(r[j]<r[index]){
                    index=j;
                }
            }
            if(index!=i){                      //将最小记录与 r[i]交换
                temp=r[i];
                r[i]=r[index];
                r[index]=temp;
            }
        }
        System.out.println("\n"+"排序好的数组为：");
        for(i=0;i<10;i++){
```

```
            System.out.print(r[i]+"   ");
        }
    }
}
```

秘笈心法

在实现选择排序时，第一将整个记录序列分为有序区和无序区，初始状态有序区为空，无序区包含所有待排序的记录；第二对无序区的比较，将使关键码小的记录与无序区的第一个记录进行交换，一直重复以上操作，直到无序区只剩下一个记录。

实例 044 插入排序法

光盘位置：光盘\MR\02\044

高级

实用指数：★★★☆

实例说明

插入排序是一类借助“插入”进行排序的方法，其主要思想是：每次将一个待排序的记录按其关键码的大小插入到一个已经排好序的有序序列中，直到全部记录排好序，本实例运行结果如图 2.28 所示。

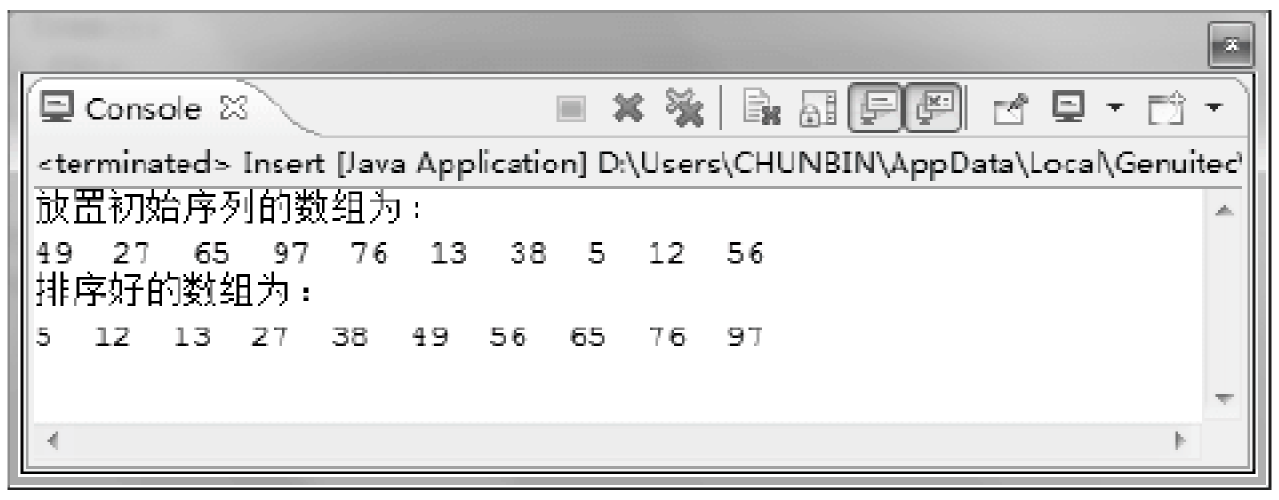


图 2.28　实例运行结果

设计过程

实现直接插入排序的关键代码如下：

```
package insert;
public class Insert {
 public static void main(String[] args) {
        int r[]={49,27,65,97,76,13,38,5,12,56};              //给出原始数的序列
        int i,j,temp,k;                                        //定义变量名称
        System.out.println("放置初始序列的数组为：");           //输出初始序列
        for(i=0;i<10;i++){
            System.out.print(r[i]+"   ");
        }
        for(i=1;i<10;i++){
            temp=r[i];
            for(j=i-1;j>=0&&temp<r[j];j--){                   //寻找插入位置
                r[j+1]=r[j];
            }
            r[j+1]=temp;                                       //大于当前值的，插到当前值后面
        }
        System.out.println("\n"+"排序好的数组为：");            //输出新序列
        for(i=0;i<10;i++){
            System.out.print(r[i]+"   ");
        }
}
}
```

秘笈心法

直接插入排序是插入排序中最简单的排序方法，首先要将待排序的记录划分为有序区和无序区，初始时有序区为待排记录的第一个记录，无序区是剩下的待排序记录。然后将无序区的第一个记录插入到有序区的适当位置，重复操作，直到无序区中没有记录。

实例 045　归并排序法

光盘位置：光盘\MR\02\045

高级

实用指数：★★★★☆

实例说明

归并排序是一种借助“归并”进行排序的方法，其实就是将两个或两个以上的有序序列合并成一个有序的序列。归并排序的主要思想是将若干有序序列逐步合并成一个有序的序列，本实例运行结果如图 2.29 所示。

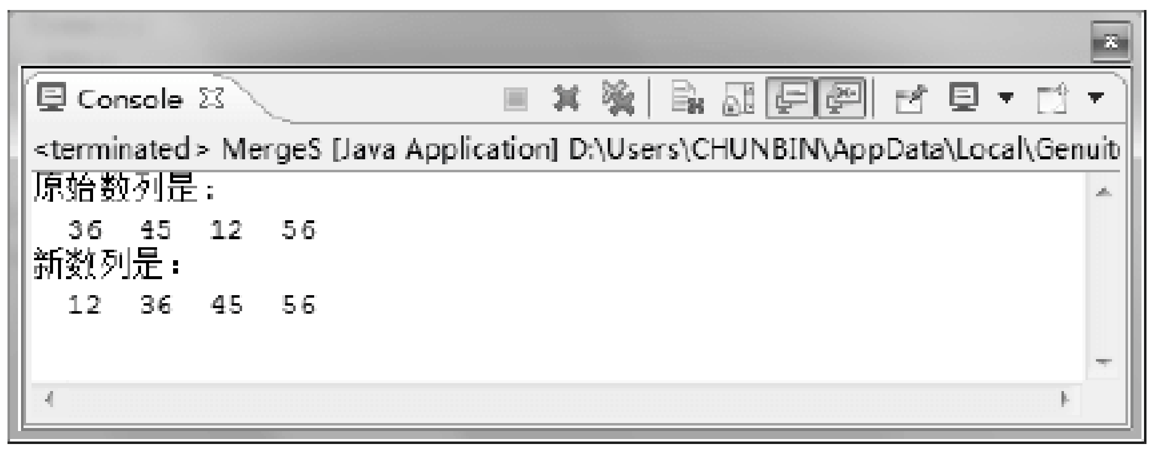


图 2.29　实例运行结果

设计过程

实现归并排序的关键代码如下：

```
package guibing;
public class MergeS {
	private static void merge(int r[],int r1[],int s,int m,int t){
		int i=s,j=m+1,k=s;
		while(i<=m&&j<=t){
			if(r[i]<=r[j]){
				r1[k++]=r[i++];			//取 r[i]和 r[j]中较小者放入 r1[k]
			}else{
				r1[k++]=r[j++];
			}
		}
		if(i<=m){
			while(i<=m){			//若第一个子序列处理完，则进行收尾处理
				r1[k++]=r[i++];
			}
		}else{
			while(j<=t){			//若第二个子序列处理完，则进行收尾处理
				r1[k++]=r[j++];
			}
		}
	}
	private static void mergePass(int r[],int r1[],int n,int h){
		int i=0;
		while(i<=n-2*h){			//待归并记录至少有两个长度为 h 的子序列
			merge(r,r1,i,i+h-1,i+2*h-1);
			i+=2*h;
		}
		if(i<n-h){
			merge(r,r1,i,i+h-1,n);		//待归并序列中有一个长度小于 h
		}else{
			for(int k=i;k<=n;k++){		//待归并序列中只剩一个子序列
				r1[k]=r[k];
			}
		}
	}
	public static void mergeS(int r[],int r1[],int n){
		int h=1;
		while(h<n){
			mergePass(r,r1,n-1,h);
			h=2*h;
			mergePass(r1,r,n-1,h);
			h=2*h;
		}
	}
	public static void main(String[] args) {
		int r[]={36,45,12,56};
		int r1[]=new int[4];
```

```
        System.out.println("原始数列是：");
        for(int i=0;i<r.length;i++){
            System.out.print("   "+r[i]);
        }
        mergeS(r,r1,r.length);
        System.out.println("\n"+"新数列是：");
        for(int i=0;i<r.length;i++){
            System.out.print("   "+r[i]);
        }
    }
}
```

秘笈心法

归并排序中的最简单的排序是二路归并排序，其主要思想是：将若干个有序的序列进行两两归并，直到所有记录全部归并到一个序列。知道了其主要思想，那么来讨论一下它的过程。将有 n 个记录的待排序列看成 n 个子序列，然后把它们两两归并，接着把长度为 2 的 n/2 个子序列再次归并，重复上述过程，直到有序为止。

2.6 算 法 应 用

实例 046　算法应用——百钱买百鸡

光盘位置：光盘\MR\02\046

高级　实用指数：★★★☆

实例说明

百钱买百鸡是一个很经典的算法案例，其主要内容是：公鸡 5 元一只，母鸡 3 元一只，小鸡一元 3 只。问 100 元钱怎样可以买 100 只鸡，本实例运行结果如图 2.30 所示。

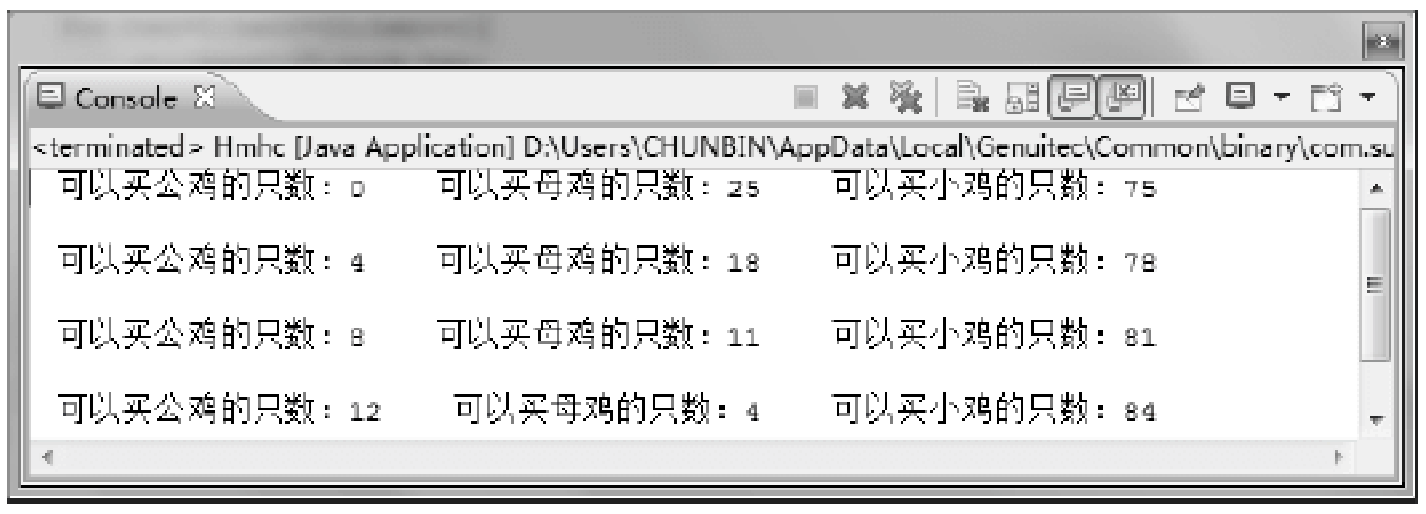


图 2.30　实例运行结果

设计过程

只要理清条件关系，就可以很容易实现。下面编写一个 Hmhc.java 将其实现，关键代码如下：

```
package hmhc;
public class Hmhc {
    public static void main(String[] args) {
        int cock,hen,chicken=0;
        for(cock=0;cock<=19;cock++){
            for(hen=0;hen<=33;hen++){
                chicken=100-cock-hen;
                int p;
                p=chicken%3;
                if(((5*cock+3*hen+chicken/3)==100)&&(p==0)){
                    System.out.print("    可以买公鸡的只数："+cock);
                    System.out.print("    可以买母鸡的只数："+hen);
                    System.out.print("    可以买小鸡的只数："+chicken);
                    System.out.println("\n");
                }
            }
        }
    }
}
```

秘笈心法

其实对于百钱买百鸡的算法，只要明白各种条件之间的关系即可。而且可以知道公鸡最多买 20 只，母鸡最多买 33 只，小鸡最多买 100 只。这样只要买各种鸡的钱总数是 100 元，鸡的只数也是 100 只即可。

实例 047　算法应用——韩信点兵

光盘位置：光盘\MR\02\047

高级

实用指数：★★★☆

实例说明

韩信点兵是一道古代数学题，其内容是：韩信带兵不足百人，3 人一行排列多一人，7 人一行排列少两人，5 人一行排列正好。本实例是计算出韩信究竟点了多少兵，实例运行结果如图 2.31 所示。

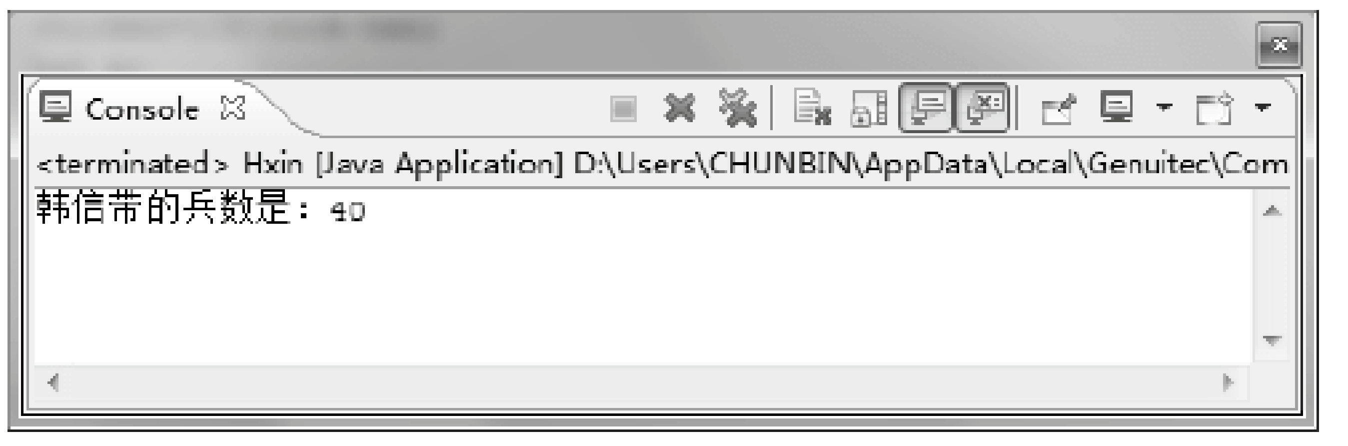


图 2.31　实例运行结果

设计过程

只要理清条件关系，就可以很容易实现。下面编写一个 Hxin.java 去实现它，实现 Hxin.java 类的关键代码如下：

```
package hanxin;
public class Hxin {
    public static void main(String[] args){
        int a=0,b=0,c=0,preson;              //定义总人数和各种站法的剩余人数
        for(preson=0;preson<100;preson++){
            a=preson%3;                      //每排 3 人的剩余人数
            b=preson%7;                      //每排 7 人的剩余人数
            c=preson%5;                      //每排 5 人的剩余人数
            if(a==1&&b==5&&c==0){            //都符合条件时的人数
                System.out.println("韩信带的兵数是："+preson);
            }
        }
    }
}
```

秘笈心法

其实对于韩信点兵的算法，只要将 7 人少两人转换为 7 人多 5 人，这样解决问题的方法就很明显了，再限定一下总的人数不多于 100 即可。

实例 048　算法应用——斐波那契数列

光盘位置：光盘\MR\02\048

高级

实用指数：★★★☆

实例说明

斐波那契数列的定义为：它的第一项和第二项均为 1，以后各项都为前两项之和。本实例将介绍如何实现斐波那契数列，实例运行结果如图 2.32 所示。

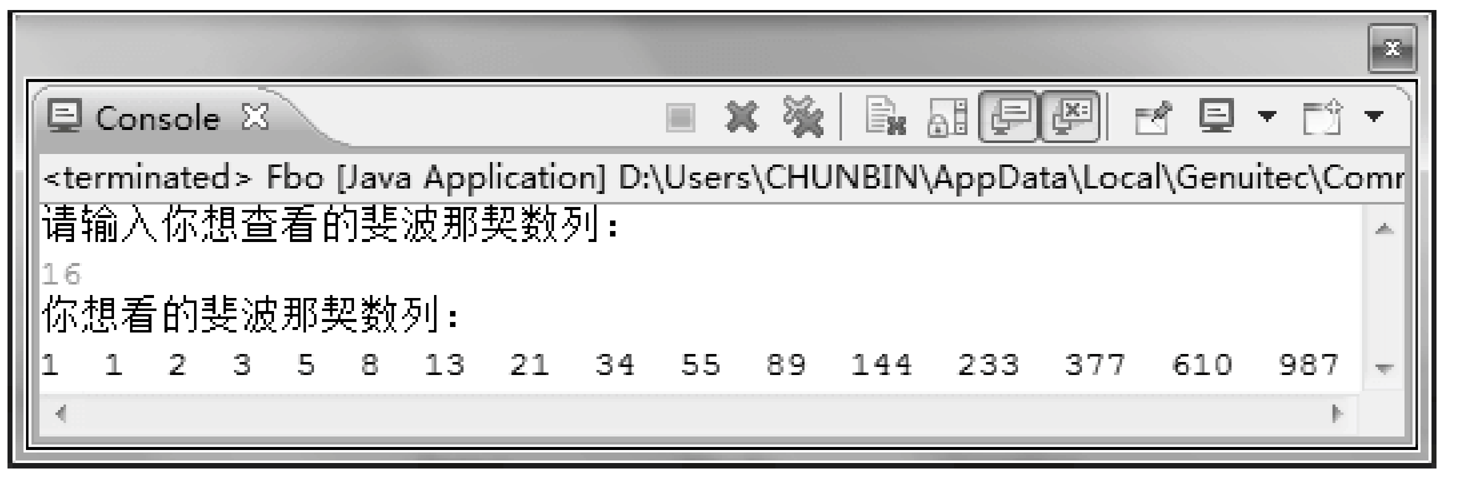


图 2.32　实例运行结果

设计过程

只要设计好循环，就可以很容易地实现。编写 Fbo.java 类，成员 f1、f2 分别是表示相邻的两项，if 条

件语句用来判定符合条件的位置，用一个 while 循环去求整个斐波那契数列，关键代码如下：

```
package fbo;
import java.util.Scanner;
public class Fbo {
private static void f(int x){
        int f1=1,f2=1,i=3;
        if(x==1)System.out.print(f1);
        if(x==2)System.out.print(f1+"    "+f2);
        if(x>=3){                                          //求位置大于 3 的数列
                System.out.print(f1+"    "+f2);
                while(x>=i){                               //求数列
                        f1=f2+f1;                          //求两项之和
                        System.out.print("    "+f1);
                        i++;
                        f2=f2+f1;
                        System.out.print("    "+f2);
                }
        }
}
public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        System.out.println("请输入你想查看的斐波那契数列：");
        int num=s.nextInt();
        System.out.println("你想看的斐波那契数列：");
        f(num/2+1);
}
}
```

秘笈心法

其实对于求斐波那契数列的算法，只要明白如何去设定一个循环即可，可以在循环中使用类似递归的赋值模式。

实例 049 算法应用——水仙花数

光盘位置：光盘\MR\02\049

高级

实用指数：★★★☆

实例说明

水仙花数是一个 3 位数，每一位数的立方相加等于该数本身。本实例将介绍如何实现水仙花数的算法，实例运行结果如图 2.33 所示。

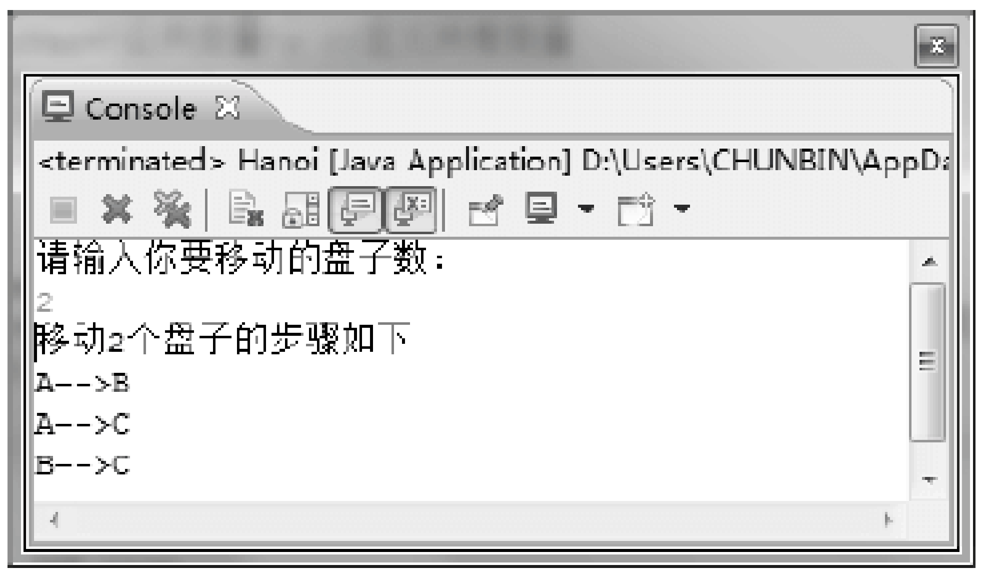


图 2.33　实例运行结果

设计过程

编写 Wflower.java 类，用于实现水仙花数的算法，关键代码如下：

```
package wf;
public class Wflower {
public static void main(String[] args) {
 int a=0,b=0,c=0;
 System.out.println("水仙花数是：");
 for (int i = 100; i < 1000; i++){                    //遍历所有 3 位数
        a = i/100;                                     //获取 3 位数中百位的数
```

```
        b=i%100/10;                          //获取 3 位数中十位的数
        c=i%100%10;                          //获取 3 位数中个位的数
        a = a * a * a;                       //计算第一位数的立方
        b = b * b * b;                       //计算第二位数的立方
        c = c * c * c;                       //计算第 3 位数的立方
        if ((a + b + c) == i)                //如果符合水仙花数
            System.out.print("    "+i);
    }
  }
}
```

秘笈心法

其实对于求水仙花数的算法，只要明白如何去设定一个循环即可。在循环中，对 100～1000 之内的数进行遍历，只要符合条件的即可进行输出。

实例 050　算法应用——素数

光盘位置：光盘\MR\02\050

高级

实用指数：

实例说明

对于素数的定义是，如果一个数只能被 1 和它本身整除，那么这个数就是素数。本实例介绍如何实现素数的算法，实例运行结果如图 2.34 所示。

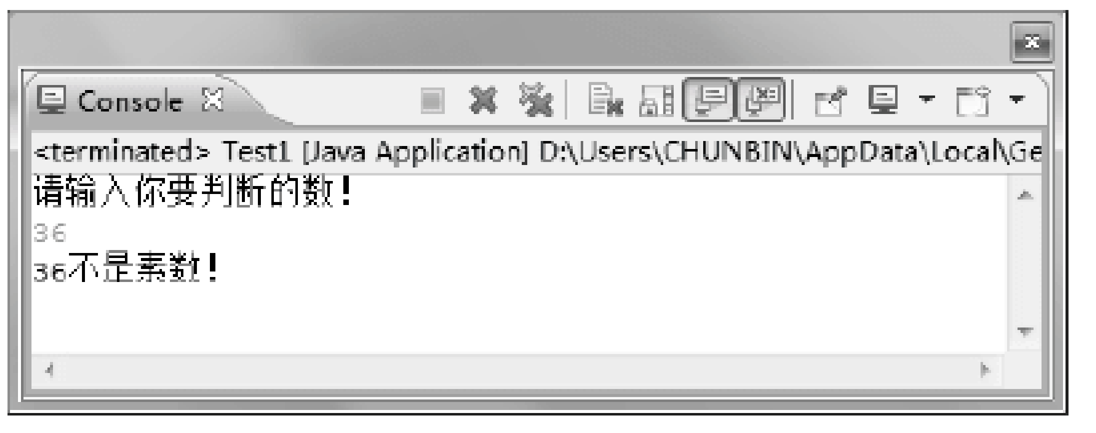


图 2.34　实例运行结果

设计过程

编写 Sushu.java 类，用于实现素数的算法，关键代码如下：

```
package sushu;
import java.util.Scanner;
public class Sushu {
 public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        System.out.println("请输入你要判断的数！");
        int x=s.nextInt();
        int i=2,flage=0;
        while(flage==0&&i<x){       //对除数进行遍历
            if(x%i==0){             //判断是否被整除
                flage=1;
            }
            else{
                i++;
            }
        }
        if(flage==0){               //对标记进行判断
            System.out.println(x+"是素数！");
        }
        else{
            System.out.println(x+"不是素数！");
        }
  }
}
```

秘笈心法

素数的算法可以用一个标记和一个 while 循环来实现。首先将标记初始化为零，用 while 循环去实现从 1 到要判断的数的前一位的遍历，每遍历一个数做一个判断，如果可以被 1 和要判断数本身之外的数整除，就将标记设置为非零的值，并跳出循环进行对标记的判断。

实例 051 算法应用——汉诺塔

光盘位置：光盘\MR\02\051

高级

实用指数：★★★☆

实例说明

汉诺塔问题是一个古典数学问题，其内容是：古代有一个梵塔，塔内有 3 个座，即 A、B、C，开始时 A 座上有 64 个盘子，盘子的大小不等，大的在下面，小的在上面。有一个老和尚想把这 64 个盘子从 A 座移到 C 座，但每次只允许移动一个盘子，且在移动过程中 3 个座上的都要保持大盘子在下面，小盘子在上面。本实例将介绍如何实现汉诺塔的算法，实例运行结果如图 2.35 所示。

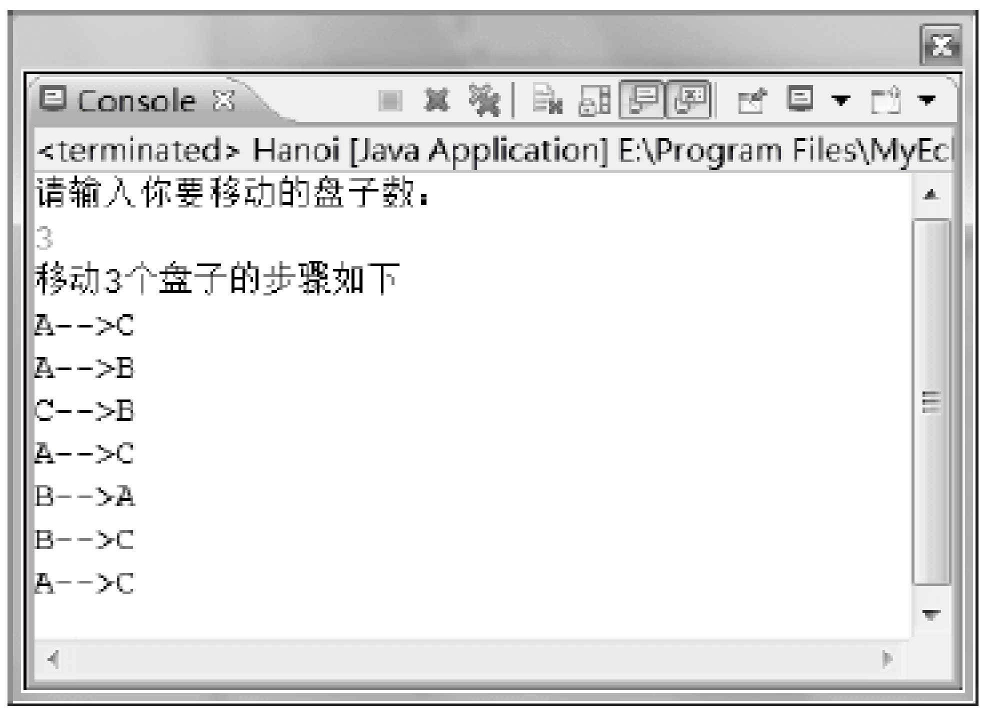


图 2.35　实例运行结果

关键技术

知道了问题，那么就来讨论一下过程：将 n 个盘子从一个座移到另一个座上，这是每个移动者要做的。除去第一个移动者外，其他都要命令其他的移动者，就是第一个移动者开始，任务层层下放，最后将一个盘子从一个座上移到另一个座上，这是第一个移动者自己做的工作。

设计过程

创建 Hanoi.java 类，编写 move()方法，将一个盘子从一个座移到另一个座；编写 hanoit()方法，参数 one、two 和 three 分别表示座一、座二和座三的变量，将盘子从 one 移到 three 座。Hanoi.java 类的关键代码如下：

```java
package digui;
import java.util.Scanner;
public class Hanoi {
 private static void move(char x,char y){
        System.out.printf("%c-->%c",x,y);
        System.out.print("\n");
 }
 //将 n 个盘子从第一座借助第二座移到第三座
 private static void hanoit(int n,char one,char two,char three){
        if(n==1){                                   //如果只有一个盘子
                move(one,three);
        }
        else{
                hanoit(n-1,one,three,two);          //将一上的盘子借助三移到二上
                move(one,three);
                hanoit(n-1,two,one,three);          //将二上的盘子借助一移到三上
        }
 }
 public static void main(String[] args) {
        int m;
        System.out.println("请输入你要移动的盘子数：");
        Scanner s=new Scanner(System.in);
        m=s.nextInt();
        System.out.println("移动"+m+"个盘子的步骤如下");
        hanoit(m,'A','B','C');
 }
}
```

秘笈心法

汉诺塔的算法是典型递归问题的算法，一定要控制好递归的条件；否则，很容易造成死循环。

第 3 章

HTML/CSS 技术

- 页面效果
- 表格样式
- 鼠标样式
- 文字及列表样式
- 文字特效
- 图片滤镜特效

3.1 页 面 效 果

在网站开发时，为了美化网站的外观，常常需要使用 CSS 样式来改变网站的整体风格，如网站的字体样式、表格样式、超链接的样式、整体颜色搭配等。下面将通过几个实例介绍在网站中常用的页面效果。

实例 052 统一站内网页风格

光盘位置：光盘\MR\03\052

初级

实用指数：★★★★

实例说明

在浏览网站时，会发现网站中所有网页的风格都相同，如页面颜色、页面字体样式、页面的背景等，这样的网站给人的感觉会很紧凑。如果把网站中的每个页面都设计成不同风格，并且制作成花花绿绿的，这样会给人造成视觉疲劳，而且整个网站显得很“散”。本实例将介绍如何应用 CSS 样式来统一网站的风格，运行结果如图 3.1 所示。

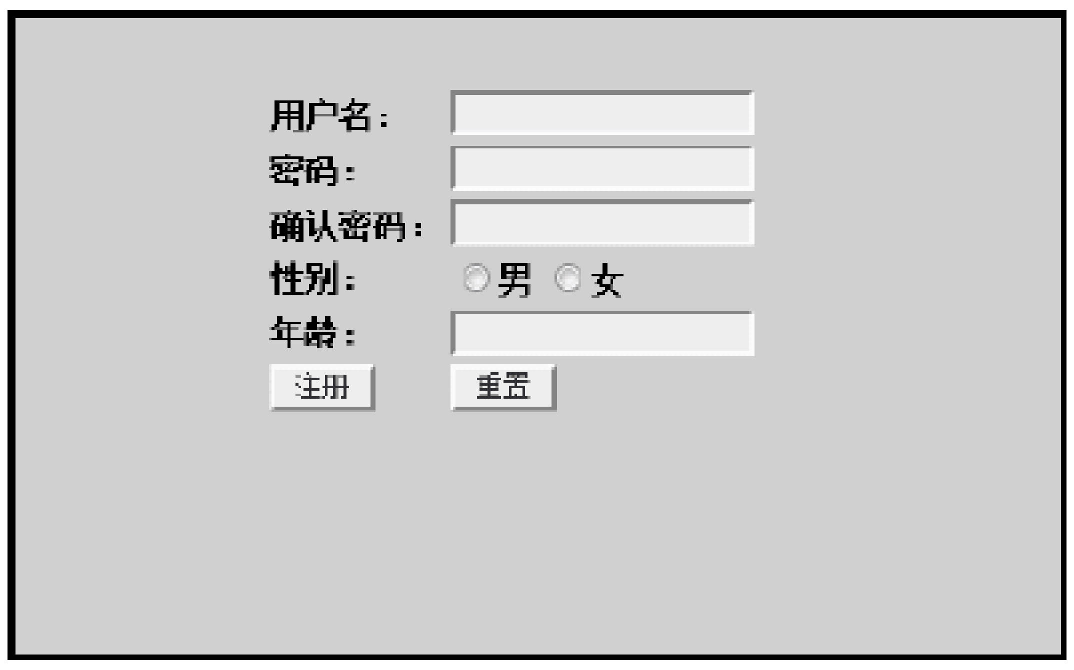


图 3.1　网站的用户注册页

关键技术

本实例主要应用 CSS 样式的字体属性、颜色和背景属性以及边框属性来实现。下面分别介绍这几种属性的用法。

1. 字体属性

字体属性包括字体的颜色、字体大小、字体的风格等，常用的字体属性及说明如表 3.1 所示。

表 3.1　字体属性及说明

字 体 属 性	说　明
font	设置或者检索对象中的文字特性的复合属性
font-family	用于指定字体名称，可以指定一个字体名，也可以用（,）分隔指定多个字体名
font-size	设置字体的字号
font-variant	设置英文大小写转换
font-weight	设置字体的粗细
font-style	用来指定是否对字体应用斜体风格

2. 颜色和背景属性

CSS 中的颜色属性用于设置页面元素的颜色，背景属性用于设置背景颜色或背景图像，具体属性及说明如表 3.2 所示。

表 3.2　颜色和背景属性及说明

颜色和背景属性	说　　明
color	设置页面的前景颜色
background-color	设置背景颜色
background-image	设置背景图像
background-repeat	设置背景图像的排列方式，包括 4 种方式，即 repeat（在水平和垂直方向上都是以瓷砖形式反复显示）、repeat-x（只在水平方向重复显示）、repeat-y（只在垂直方向重复显示）和 no-repeat（不重复只显示一个）
background-attachment	设置背景图像是否固定，fixed（固定背景图像）、scroll（背景图像和其他内容一起滚动）
background-position	设置背景图像的显示位置
background	综合设置背景图像的属性

3. 边框属性

边框属性用于设置元素的边框宽度、样式和颜色，具体属性及说明如表 3.3 所示。

表 3.3　边框属性及说明

边 框 属 性	说　　明
border	边框复合属性
border-top	上边框
border-left	左边框
border-right	右边框
border-bottom	下边框
border-color	边框颜色
border-style	边框样式
border-width	边框宽度
border-top-color	上边框颜色
border-left-color	左边框颜色
border-right-color	右边框颜色
border-bottom-color	下边框颜色
border-top-style	上边框样式
border-left-style	左边框样式
border-right-style	右边框样式
border-top-width	上边框宽度
border-left-width	左边框宽度
border-right-width	右边框宽度
border-bottom-width	下边框宽度

边框样式的属性及说明如表 3.4 所示。

表 3.4　边框样式的属性及说明

边框样式属性	说　　明
none	无边框
hidder	隐藏边框 IE 不支持
dotted	边框由点组成
dashed	边框由短线组成

续表

边框样式属性	说　明
solid	边框是实线
double	边框是双实线
groove	边框带有立体感的沟槽
ridge	边框成脊形
inset	边框内嵌一个立体边框
outset	边框外嵌一个立体边框

设计过程

（1）定义 CSS 样式表文件，将其命名为 mycss.css，在该文件中使用 body 设置页面的样式，使用 td 设置所有单元格内的字体样式，使用 input 设置所有文本框的样式。其中.td1 是设置具体某个单元格内的字体样式，.botton1 是设置按钮的显示样式，关键代码如下：

```
body{
 margin:2em;
      background-color: #CCCCFF;
}
td {
 font-family: "Times New Roman", Times, serif;
 font-size: 14px;
        font-weight: bold;
}
input {
        font-size:9pt;
        color: #003399;
        font-family: 宋体;
        border: '1px' 'dashed' '#999999';
        background: #EEEEEE;
}
.td1{
        font-size: 12pt;
        font-family: 黑体;
}
.botton1{
        background: #FFFFFF;
        font-weight:bold;
        width: 50px;
}
```

（2）新建 index.html 网页，在页面中引用外部样式表文件 mycss.css，粗体代码部分为具体引入的 CSS 文件，具体代码如下：

```
  <head>
    <title>统一站内风格</title>
      <link rel="stylesheet" type="text/css" href="mycss.css">
  </head>
```

（3）在页面中引用相应的 CSS 样式，关键代码如下：

```
<body>
  <form action="">
      <table align="center">
            <tr>
                  <td>用户名：</td><td><input type="text" name="name" /></td>
            </tr>
            <tr>
                  <td >密码：</td><td><input type="password" name="pwd" /></td>
            </tr>
            <tr>
                  <td >确认密码：</td><td><input type="password" name="pwd1" /></td>
            </tr>
            <tr>
                  <td >性别：</td>
```

```
                <td>
                        <input type="radio" name="sex" value="m"/>男
                        <input type="radio" name="sex" value="f"/>女
                </td>
        </tr>
        <tr>
                <td >年龄：</td><td><input type="text" name="age" /></td>
        </tr>
        <tr>
                <td><input class="botton1" type="submit" value="注册" /></td>
                <td><input class="botton1" type="button" value ="重置"></td>
        </tr>
    </table>
  </form>
</body>
```

秘笈心法

一个网站可能有不同的风格样式，而且网站中不同的元素可能风格也不同。为了实现不同风格样式的切换，可以定义多个不同风格样式的 CSS 样式文件，在网站需要使用时，直接通过<link>标签导入即可。

实例 053　设置超链接文字的样式

光盘位置：光盘\MR\03\053

初级

实用指数：★★★☆

实例说明

默认情况下，使用<a>标签定义的超链接，字体颜色为蓝色，可以通过 CSS 样式来改变链接字体的颜色以及样式。运行本实例，当鼠标经过超链接文字时，将文字颜色改变并将字体改为粗体；当鼠标移出时，文字改为初始状态，运行结果如图 3.2 所示。

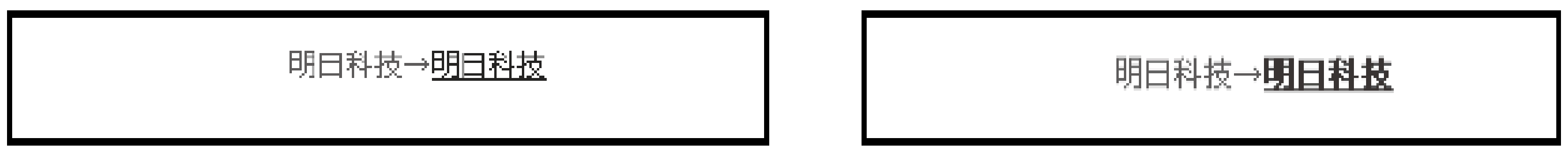


图 3.2　超链接的显示样式

关键技术

在 CSS 样式中，对超链接的样式有以下几种定义。

（1）设置链接未被访问时的样式，具体语法如下：

```
a:link{font-size:10px;... }
```

（2）设置链接在鼠标经过时的样式，具体语法如下：

```
a:hover{font-size:10px;text-decoration:underline;color:#ff0000}
```

（3）设置链接激活时的样式，具体语法如下：

```
a:active{font-size:10px;...}
```

（4）设置链接已被访问过的样式，具体语法如下：

```
a:visited{font-size:10px;color:#00ffff;...}
```

设计过程

（1）新建 index.html 页，在该页中首先定义超链接的 CSS 样式，关键代码如下：

```
<head>
    <title>超链接的样式</title>
      <meta http-equiv="pragma" content="no-cache">
      <meta http-equiv="cache-control" content="no-cache">
      <meta http-equiv="expires" content="0">
      <style type="text/css">
            td{
                  font-size:9pt;
                  color:#007766;
            }
```

```
            a:link{
                    font-size:9pt;
                    color:#cccccc;
                    font-weight:bold;
                    text-decoration: underline;
            }
            a:hover{
                    font-size:9pt;
                    color:#ff0000;
                    font-weight:bold;
                    text-decoration: underline;
            }
            a:active{
                    font-size:9pt;
                    color:#6699ff;
                    font-weight:bold;
                    text-decoration: underline;
            }
        </style>
</head>
```

（2）在该页中使用<a>标签添加超链接，关键代码如下：

```
<table align="center">
        <tr>
                <td>明日科技→<a href="#">明日科技</a></td>
        </tr>
</table>
```

秘笈心法

超链接是每个网站中必有的功能，换句话说它是网站中最常用的功能。所以读者有必要掌握在 CSS 样式中超链接的几个属性（a:link、a:hover、a:active、a:visited）的应用。

实例 054 网页换肤

光盘位置：光盘\MR\03\054

初级

实用指数：★★★★

实例说明

网页换肤的功能在网站上应用得非常广泛，用户可以根据自己的爱好，选择不同的页面风格。运行本实例，如图 3.3 所示，当单击网页中的“橘色经典”链接时，页面风格将变为橘色主题；当单击“灰色畅想”链接时，页面风格将变为灰色主题。

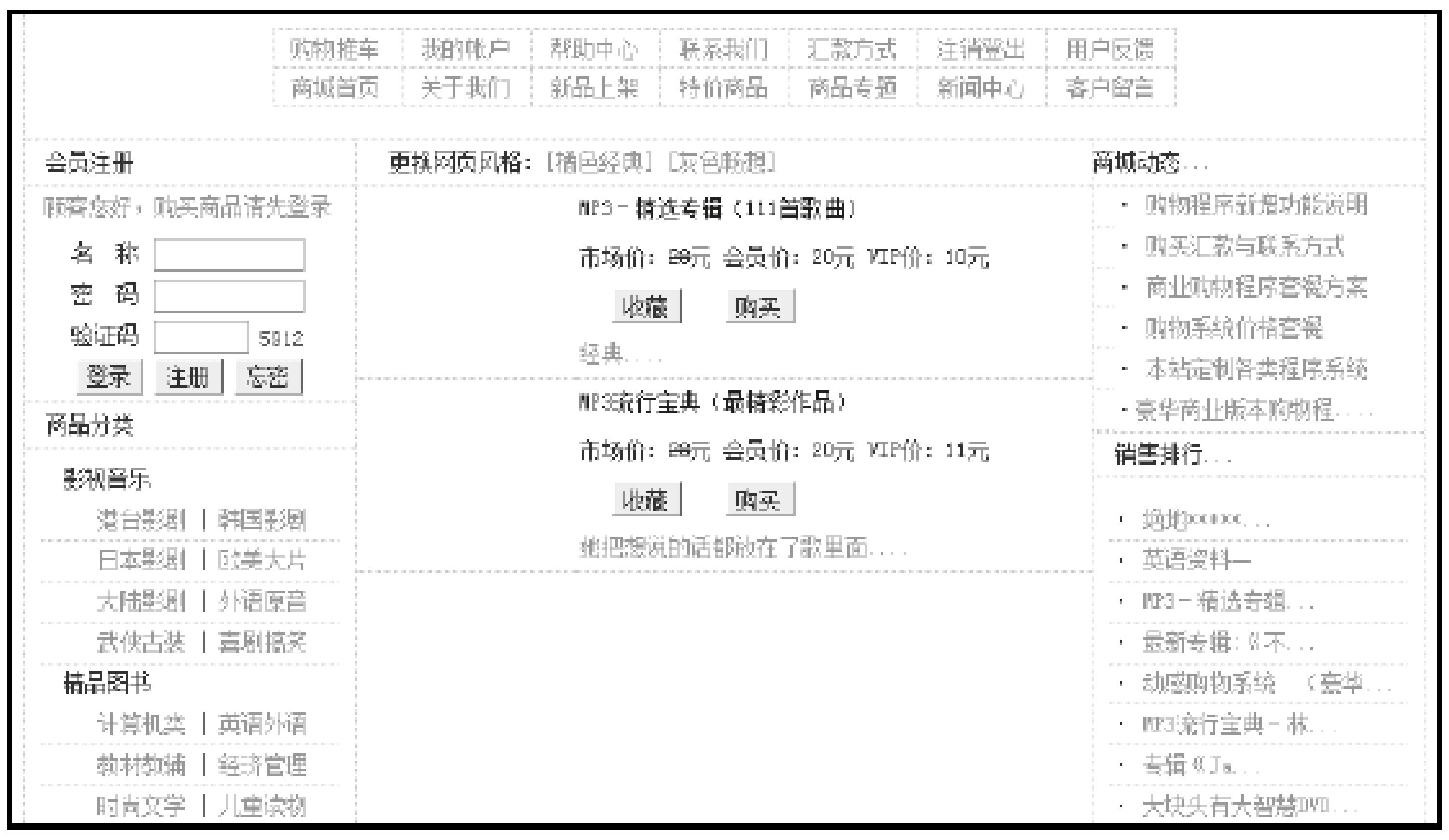


图 3.3　可以更换网页风格的页面

关键技术

实现换肤功能，主要是根据单击超链接时，调用 JavaScript 方法来动态改变页面的<link>标签的 href 属性值实现的。首先需要事先定义好两个不同风格的 CSS 样式文件，然后单击某个风格的链接动态实现改变。

设计过程

（1）编写不同风格的 CSS 样式文件，详细代码请参见本书附带的光盘。

（2）新建 index.htm 网页，在页面中引用其中一个 CSS 样式文件作为网页默认的样式，关键代码如下：

```
<link id="myCss" href="orange.css" rel="stylesheet">
```

（3）在<script>标签中编写保存 cookie 信息的方法，用于将 CSS 样式文件的路径信息保存到客户端 Cookie 文件中，具体代码如下：

```
function writeCookie(csspath){
	var today = new Date();
	var expires = new Date();
	expires.setTime(today.getTime() + 1000*60*60*24*30);	//有效期为 30 天
	var str="cssPath="+csspath+";expires=" + expires.toGMTString()+";" ;
	document.cookie=str;
}
```

（4）编写读取 cookie 信息的方法，关键代码如下：

```
function readCookie(cookieName){
	var search = cookieName + "=";
	if (document.cookie.length > 0) {
		offset = document.cookie.indexOf(search);
		if (offset != -1) {
			offset += search.length;
			end = document.cookie.indexOf(";", offset);
			if (end == -1){
				end = document.cookie.length;
				return unescape(document.cookie.substring(offset, end));
			}
		}
	}
}
```

（5）编写超链接的 onClick 事件调用的方法，关键代码如下：

```
function change(type){
	if(type=="orange"){
		document.getElementById("myCss").href="orange.css";
		writeCookie("orange.css");
	}
	if(type=="gray"){
		document.getElementById("myCss").href="gray.css";
		writeCookie("gray.css");
	}
}
```

（6）在页面中超链接的关键代码如下：

```
<a href="#" onClick="change('orange')">[橘色经典]</a>
<a href="#" onClick="change('gray')">[灰色畅想]</a>
```

秘笈心法

在实现网页换肤时，需要将每次执行所调用的 CSS 样式文件的路径信息保存到客户端的 cookie 中。为什么呢？因为当用户单击链接换肤后，网页确实是换了一种风格，但是随着浏览器的关闭，网页改变的风格将会消失，因为浏览器并不会自动保存用户所执行的这次改变操作，当再次打开浏览器访问此页时，网页风格还是没改变之前的，所以需要使用 JavaScript 将每次的操作都保存在 cookie 中。

实例 055	滚动文字 光盘位置：光盘\MR\03\055	初级 实用指数：★★★★

实例说明

在许多网站中，都会包含滚动文字的特效，这些特效也可以称为走马灯特效，如一些网站中的新闻公告。

本实例将介绍如何在网页中实现滚动文字。运行本实例，如图 3.4 所示，将从网页的下方向上逐渐出现滚动的文字信息。

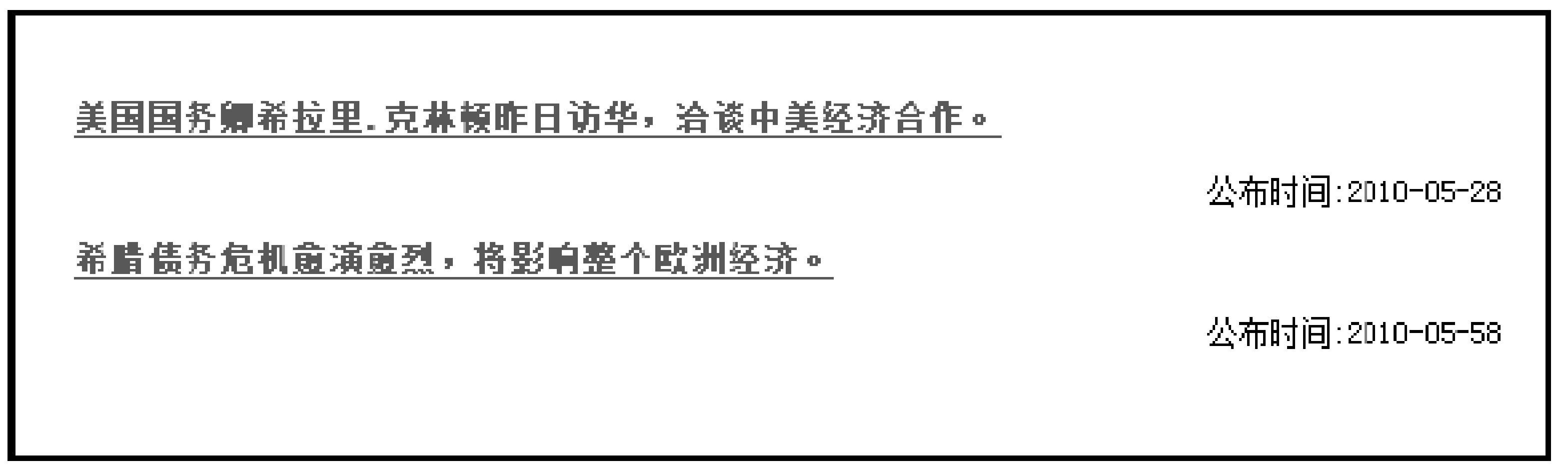


图 3.4　在网页中添加滚动文字

关键技术

实现滚动文字的特效，主要是在页面中应用<marquee>标签。该标签专门用于实现文字或图片的滚动效果，其常用属性及说明如表 3.5 所示。

表 3.5　<marquee>标签的属性及说明

<marquee>标签的属性	说　明
align	滚动内容的对齐方式：top（顶端对齐）、middle（居中对齐）、bottom（向下对齐）
direction	滚动的方向：up（向上）、down（向下）、left（向左）、right（向右）
behavior	滚动的方式：scroll（循环滚动）、slide（一次滚动）、alternate（交替滚动）
loop	循环滚动的次数，取值为-1 或 Infinite 表示无限次滚动
scrollamount	滚动的速度，单位为像素，值越大滚动速度越大
scrolldelay	两次滚动的间隔时间
width	滚动区域的宽度，单位为像素或以百分比形式表示
height	滚动区域的高度
bgcolor	滚动区域的背景颜色
hspace	滚动区域与浏览器边界的水平距离
vspace	滚动区域与浏览器边界的垂直距离

设计过程

创建 index.htm 页面，在该页中的适当位置将要滚动的内容添加到<marquee>标签中，然后通过设置<marquee>标签的相关属性来自定义滚动的显示效果，关键代码如下：

```
<marquee direction="up" onMouseOver="this.scrollAmount='1'"onMouseOut="this.scrollAmount='2'" onMouseDown="this.scrollAmount='4'; this.direction='down'"onMouseUp="this.scrollAmount='1';this.direction='up'" scrollAmount="2" height="291">
  <table cellspacing="2" cellpadding="0" border="0" width="100%" align="center">
 <tr>
       <td height="30" style="color:yellow;font-size:10pt;font-weight:bold;">
       <a href="#">美国国务卿希拉里·克林顿昨日访华，洽谈中美经济合作。</a>
       </td>
 </tr>
 <tr>
       <td height="20" align="right">公布时间:2010-05-28  </td>
 </tr>
       <tr>
         <td height="30" style="color:yellow;font-size:10pt;font-weight:bold;">
              <a href="#">希腊债务危机愈演愈烈，将影响整个欧洲经济。</a>
         </td>
       </tr>
       <tr>
         <td height="20" align="right"> 公布时间:2010-05-58 </td>
       </tr>
    </table>
</marquee>
```

秘笈心法

通过<marquee>标签的属性可以实现不同的滚动效果，其中最常用的属性是滚动的方向（direction）、滚动的方式（behavior）以及滚动的速度（scrollamount）。

实例 056　制作渐变背景　初级

光盘位置：光盘\MR\03\056

实用指数：★★★★

实例说明

在浏览网站时，常常会发现有些网站页面背景有渐变的效果，这种效果可以使页面更加美观。本实例将介绍如何实现网页的渐变背景，实例运行效果如图 3.5 所示，页面背景是渐变的效果。

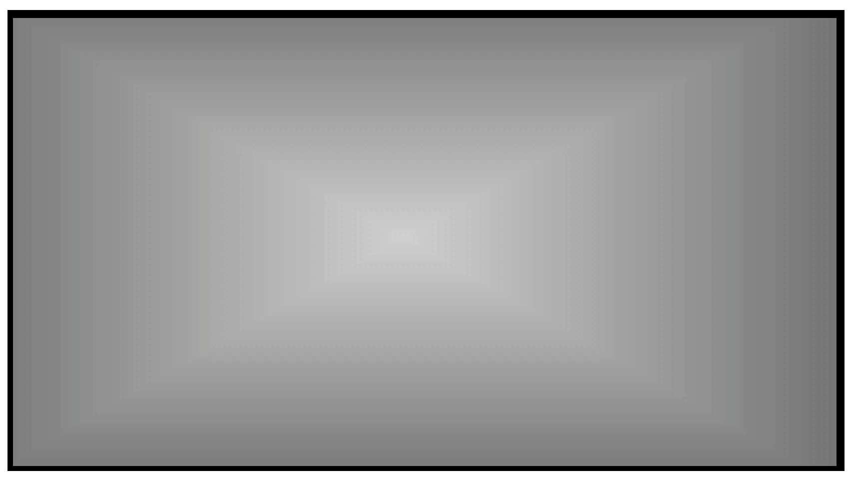


图 3.5　具有渐变背景的网页

关键技术

实现网页背景的渐变，主要是通过在 CSS 样式中定义 Alpha 滤镜来实现的。在 Alpha 滤镜中，需要设置渐变的属性值，其主要的属性及说明如表 3.6 所示。

表 3.6　Alpha 滤镜的属性及说明

属　　性	说　　明
Opacity	表示透明水准，默认范围是 0～100，0 表示完全透明，100 表示完全不透明
Finishopacity	可选属性，设置渐变的透明效果时，使用该参数指定结束时的透明度，范围在 0～100 之间
Style	指定透明区域的形状特征：0（统一形状）、1（线形）、2（放射状）、3（长方形）
Startx	渐变透明效果的开始 x 坐标
Starty	渐变透明效果的开始 y 坐标
Finishx	渐变透明效果的结束 x 坐标
Finishy	渐变透明效果的结束 y 坐标

设计过程

新建 index.htm 网页，在该页中的<style>标签中加入渐变效果的 CSS 样式，具体代码如下：

```
<style type="text/css">
body{
        background:green;
        filter: Alpha(Opacity=20,Finishopacity=80,Style=3,Startx=100,Starty=100,Finishx=0,Finishy=0);
}
</style>
```

秘笈心法

为了提高开发效率，可以将 Alpha 滤镜的样式代码写到.css 样式文件中，在不同网页中应用此样式时直接通过<link>标签引用 CSS 文件即可。

实例 057 CSS 控制绝对定位

光盘位置：光盘\MR\03\057

初级

实用指数：★★★★

实例说明

在网页布局的大部分情况下需要使用表格嵌套来实现定位，这种方法虽然可以在一定程度上解决问题，但是它却有个致命的缺点，就是如果表格中的某个元素的位置发生改变就有可能打乱整个页面的布局。本实例将介绍如何应用 CSS 样式控制页面的绝对定位，其实现的效果示意图如图 3.6 所示。

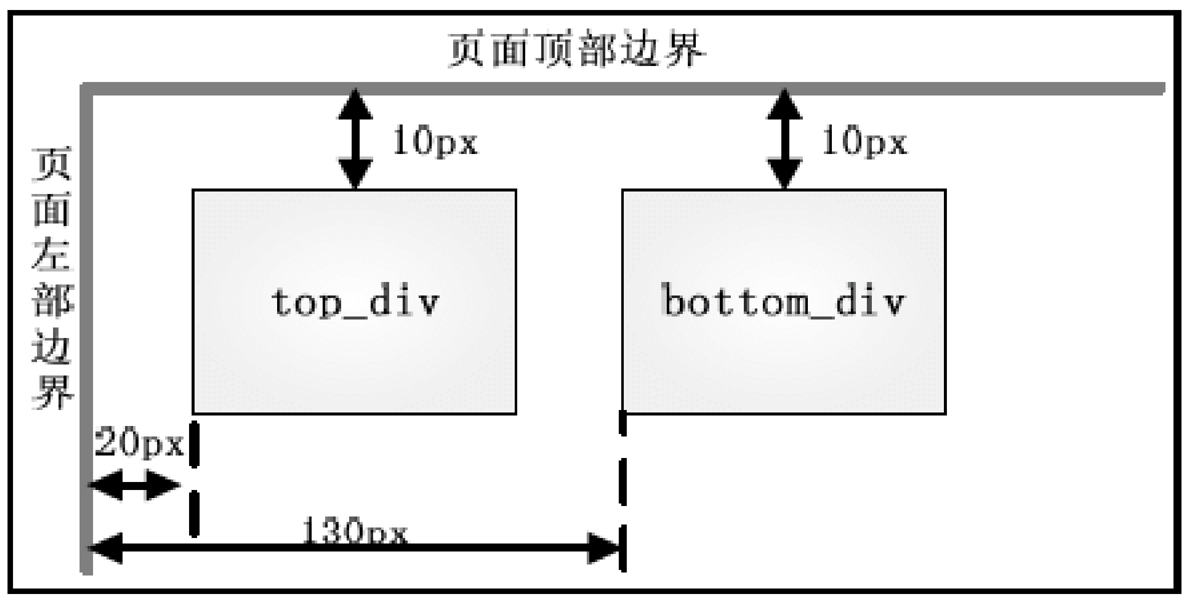


图 3.6　CSS 的绝对定位显示效果

关键技术

在 CSS 中提供了更加灵活的定位方法，常用的定位属性包括 position、left、top、width、height 等，其中 position 属性就是实现绝对定位的关键属性，它采用了 3 种定位方式，包括绝对定位（absolute）、相对定位（relative）和静态定位（static）。

设计过程

新建 index.htm 网页，利用 CSS 实现页面中两个图片的绝对定位，将两个图片分别放置到 div 中并为 div 定义两个不同的 id，第一张图片定义的 id 值为 top_Div，第二张图片定义的 id 值为 bottom_Div，其中 CSS 属性设置的关键代码如下：

```
#top_Div {
        position: absolute;                /*绝对定位*/
        left: 20px;                        /*距离左侧页面 20px*/
        top: 10px;                         /*距离顶部页面 10px*/
}
#bottom_Div {
        position: absolute;                /*绝对定位*/
        left: 130px;                       /*距离左侧页面 130px*/
        top: 10px;                         /*距离顶部页面 10px*/
}
```

秘笈心法

绝对定位使元素的位置与文档流无关，因此不占据空间，这一点正好与相对定位不同，相对定位实际上被看作普通流定位模型的一部分，因为元素的位置是相对于它在普通流中的位置。

实例 058 CSS 控制垂直居中

光盘位置：光盘\MR\03\058

初级

实用指数：★★★★

实例说明

HTML 页面居中的需求很常见，在实际的项目应用中也会经常碰到类似的问题。但是利用 CSS+DIV 的方

法来实现更为简单，只要 5～6 行代码即可实现。本实例将介绍如何利用 CSS 来设置页面的垂直居中效果，实现效果如图 3.7 所示。



图 3.7　CSS 控制的文字垂直居中显示效果

关键技术

实现垂直居中关键的几个属性是 text-align、line-height 和 vertical-align。text-align 属性表示文字的对齐样式，line-height 属性表示垂直居中的高度，vertical-align 属性表示垂直居中。

设计过程

新建 index.htm 网页，首先将需要居中显示的内容用 div 包裹起来，并为 div 定义一个 id 值 content，然后在 CSS 文件中定义该 div 的 CSS 属性，关键代码如下：

```
#content {
        text-align: center;                    /*文字水平居中*/
        margin-right: auto;
        margin-left: auto;
        vertical-align: middle;                /*文字垂直居中*/
        line-height: 200px;                    /*垂直居中的高度*/
        height: 200px;                         /*div 的高度*/
        width: 400px;                          /*div 的宽度*/
        background: #cef;                      /*div 的背景颜色*/
}
```

秘笈心法

利用 CSS 实现垂直居中时，必须是 line-height 和 vertical-align 两个属性搭配使用，否则达不到预期的效果。

实例 059	CSS 实现的图文混排 光盘位置：光盘\MR\03\059	初级 实用指数：★★★★

实例说明

现在在大部分的网站中都可以看到图文混排效果的页面布局，那么这种效果又是如何实现的呢？本实例将介绍如何利用 CSS 实现图文混排效果，实例运行效果如图 3.8 所示，图在左边，文字在图的右边和下方。

关键技术

本实例的实现主要是利用 CSS 的 float 属性，应用该属性可以完成图文混排，该属性包含 4 个可选值，分别为 left、right、none 和 inherit。



图 3.8　设置图文混排后的效果

设计过程

（1）新建 index.html 页，利用 CSS 实现图文混排的页面布局效果，CSS 设置的关键代码如下：

```
.picture {
        float: left;                          /*在文本的左边*/
        border: 1px solid #000000;            /*图片边框的颜色*/
        margin-top: 10px;                     /*上方与其他元素保持 10px*/
        margin-bottom: 10px;                  /*下方与其他元素保持 10px*/
        margin-left: 10px;                    /*左侧与其他元素保持 10px*/
        margin-right: 10px;                   /*右侧与其他元素保持 10px*/
}
```

（2）在图片中调用这个 CSS 类，实现图文混排的页面效果，关键代码如下：

```
<img src="head.png" class="picture"/>
```

秘笈心法

利用 CSS 样式中的 float 属性即可实现图文混排效果，如果再配合一些其他属性的使用，就可以达到更加完美的效果。

3.2　表 格 样 式

在设计网站时，对表格的应用非常频繁。可以使用表格布局整个网页，也可以在网页中显示列表信息等。在使用表格时，可以使用 CSS 样式对表格的样式进行设置。下面将介绍几个典型的通过 CSS 样式来设置表格的例子。

实例 060	只有外边框的表格 光盘位置：光盘\MR\03\060	初级 实用指数：★★★☆

实例说明

在浏览网页时，经常会看到网页中的表格只有外边框，而表格内容没有边框。本实例将通过 CSS 样式来实现表格只有外边框的样式，实例运行效果如图 3.9 所示，整个网页是用表格制作的，而且只显示表格的外边框。

关键技术

实现本实例主要是应用 CSS 样式的边框复合属性 border，border 表示表格的边框属性。该属性在实例 052 中已经介绍，此处不再具体讲解。



图 3.9　只有外边框的表格

设计过程

（1）新建 index.htm 网页，在页面的<style>标签中编写 CSS 样式，具体代码如下：

```
<style type="text/css">
        .tableBorder{
                border:1px solid #407D2A;
        }
</style>
```

（2）在页面的表格<table>标签中，应用以上定义的 CSS 样式，具体代码如下：

```
<body>
<table class="tableBorder" align="center">
        <tr>
                <td>用户名：</td>
                <td><input type="text" name="name" /></td>
        </tr>
        <tr>
                <td>密码：</td>
                <td><input type="password" name="pwd" /></td>
        </tr>
        <tr>
                <td>确认密码：</td><td><input type="password" name="pwd1" /></td>
        </tr>
        <tr>
                <td>年龄：</td><td><input type="text" name="age" /></td>
        </tr>
        <tr>
                <td>性别：</td>
                <td><input type="radio" name="sex" value="m"/>男<input type="radio" name="sex" value="f"/>女</td>
</tr>
        <tr><td><input type="submit"   value="注  册"/></td><td><input type="reset"   value="重  置"/></td></tr>
</table>
</body>
```

秘笈心法

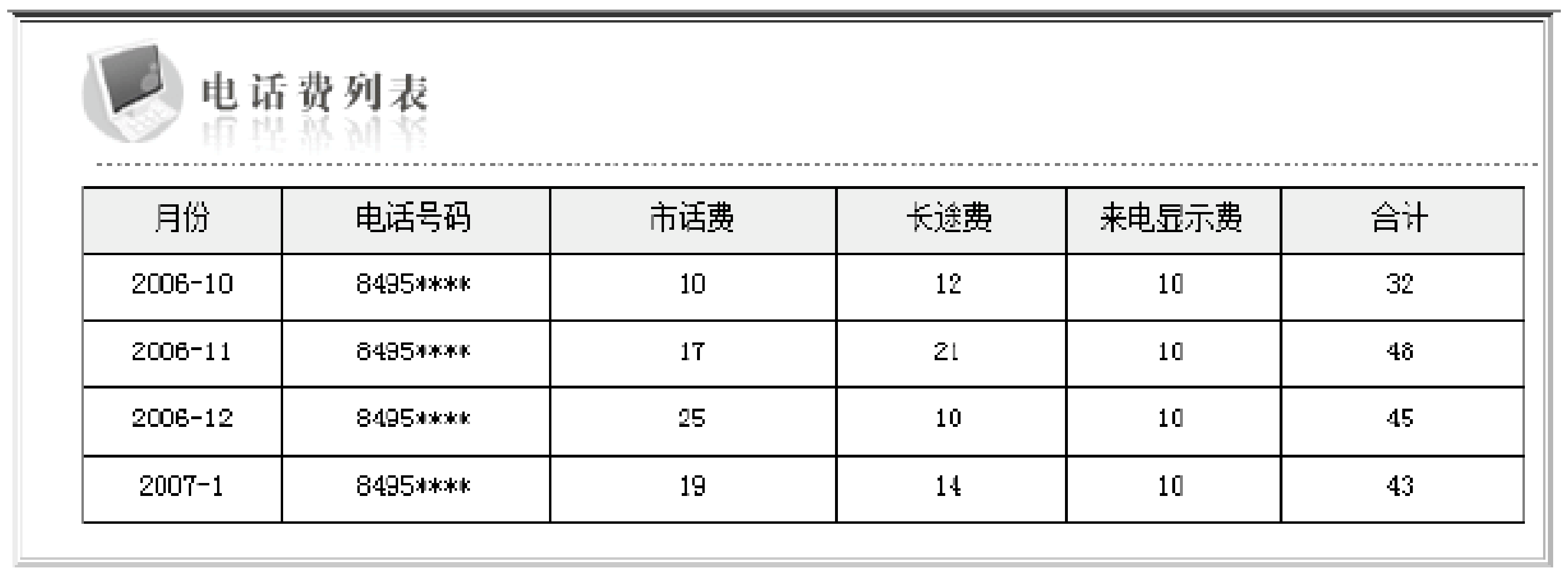
除了应用边框复合属性 border 以外，还可以应用 border-width 属性、border-color 属性以及 border-style 属性来设置表格的边框样式。

实例 061	彩色外边框的表格 光盘位置：光盘\MR\03\061	初级 实用指数：★★★☆

实例说明

在制作网站时，为了使网站中显示数据的表格更加醒目，可以将表格四周的边框设置成不同颜色。本实例

通过使用 CSS 样式来设置表格的外边框颜色，实例运行效果如图 3.10 所示。


电话费列表

月份	电话号码	市话费	长途费	来电显示费	合计
2006-10	8495****	10	12	10	32
2006-11	8495****	17	21	10	48
2006-12	8495****	25	10	10	45
2007-1	8495****	19	14	10	43

图 3.10　彩色外边框的表格

关键技术

本实例主要是通过 CSS 的边框颜色属性设置表格四周的边框为不同颜色而实现的，可以在 CSS 样式中使用以下边框的颜色属性。

- ❑ border-left-color：左边框颜色。
- ❑ border-right-color：右边框颜色。
- ❑ border-top-color：上边框颜色。
- ❑ border-bottom-color：下边框颜色。

设计过程

（1）新建 index.htm 网页，在页面的<style>标签中编写表格边框的 CSS 样式，具体代码如下：

```
<style type="text/css">
        .table1{
                border-bottom-color:#00FFFF;
                border-left-color:#FFFF00;
                border-top-color:#FF0000;
                border-right-color:#00FF00;
                border-style:double;
        }
</style>
```

（2）在页面的表格<table>标签中，应用以上定义的 CSS 样式，关键代码如下：

```
<table width="643"   border="0" align="center" cellpadding="0" cellspacing="0" class="table1">
```

秘笈心法

在设置表格边框颜色时，除了使用具体的 RGB 颜色值以外，还可以使用英文来表示某个颜色值。如红色为 red，黄色为 yellow，绿色为 green，橘红为 orangered 等。

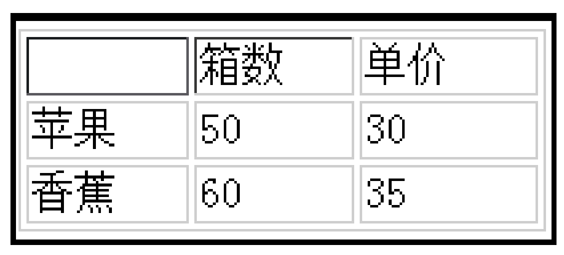
实例 062　单元格的边框变色

光盘位置：光盘\MR\03\062

初级　实用指数：★★★☆

实例说明

在设计制作网站时，为了达到预期效果，有时会在表格中加入一些特效。运行本实例，如图 3.11 所示，当鼠标经过某个单元格时，此单元格边框变色；鼠标移出单元格时，边框颜色恢复初始状态。



	箱数	单价
苹果	50	30
香蕉	60	35

图 3.11　单元格的边框变色

关键技术

实现本实例，需要编写两个鼠标事件的方法，分别是 onMouseOver（鼠标经过事件）的方法和 onMouseOut（鼠标移出事件）的方法。在方法中，通过鼠标事件获得某个单元格对象，然后通过改变其 style 属性来实现改

变边框颜色。如某个单元格的 id 属性值为 td1，修改它的左边框颜色的代码如下：

```
document.getElementById("td1").style.borderLeftColor="0000FF";
```

设计过程

（1）新建 index.htm 网页，在页面的<script>标签中编写鼠标经过事件的方法，关键代码如下：

```
function over(id){
        document.getElementById(id).style.borderLeftColor="green";
        document.getElementById(id).style.borderRightColor="green";
        document.getElementById(id).style.borderTopColor="green";
        document.getElementById(id).style.borderBottomColor="green";
}
```

（2）编写鼠标移出事件的方法，关键代码如下：

```
function out(id){
        document.getElementById(id).style.borderLeftColor="pink";
        document.getElementById(id).style.borderRightColor="pink";
        document.getElementById(id).style.borderTopColor="pink";
        document.getElementById(id).style.borderBottomColor="pink";
}
```

（3）在页面的<td>元素中调用以上两个方法，关键代码如下：

```
<td width="60" id="Ttd0" onmouseover="over('Ttd0')" onmouseout="out('Ttd0')"> </td>
```

秘笈心法

根据本实例，可以实现表格的行变色。主要是在表格的<tr>标签中使用 onMouseOver 事件以及 onMouseOut 事件来调用 JavaScript 方法，然后在该方法中通过表格的 id 属性获得表格的行对象，最后控制行对象的属性即可。

实例 063	表格外边框具有霓虹灯效果 光盘位置：光盘\MR\03\063	初级 实用指数：★★★☆

实例说明

有些网站为了更多地吸引浏览者注意，会在网站中加入具有霓虹灯特效的表格。本实例将通过调用 JavaScript 方法来实现表格外边框的霓虹灯效果。本实例的运行效果如图 3.12 所示，表格的外边框会定时变色，从而达到霓虹灯的效果。

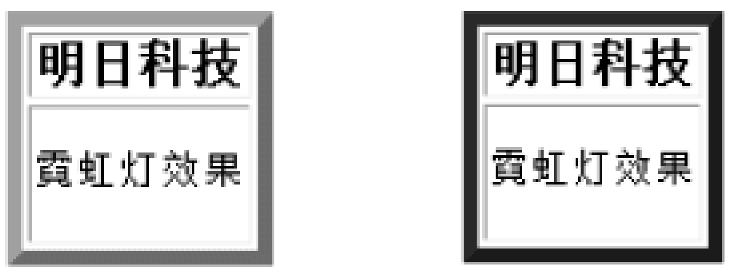


图 3.12 表格的外边框具有霓虹灯效果

关键技术

要实现不断地改变表格边框颜色，可以通过 JavaScript 中提供的 setTimeout()方法，该方法会按照指定的参数时间来调用自定义的 JavaScript 方法，该方法的语法格式如下：

```
setTimeout(funtion,milliseconds)
```

参数说明

❶ function：要调用的自定义函数名称。

❷ milliseconds：设置超时时间，以毫秒为单位。

设计过程

（1）新建 index.htm 网页，在页面中添加一个表格，关键代码如下：

```
<table id="table1" border=5>
        <tr>
```

```
            <td align=center><strong>明日科技</strong></td>
        </tr>
        <tr>
            <td align=center><pre>霓虹灯效果</pre></td>
        </tr>
</table>
```

（2）编写 JavaScript 方法，实现表格边框变色，关键代码如下：

```
<script language="javascript">
var i=0;
var Color= new Array("#0000FF","#99FF00","#660033","#CC66CC","#FFFF33");//定义颜色值的数组
function change(){
        if (i>Color.length-1) i=0;              //如果 i 的值大于数组的元素数，则将 i 的值改为 0
        table1.style.borderColor=Color[i];      //改变表格边框颜色
        i+=1;
        setTimeout("change()",500);             //每隔 500 毫秒调用一次该方法
}
</script>
```

（3）在页面加载时，调用该方法实现变色，主要设置页面的 onload 事件，关键代码如下：

```
<body onload="change()">
```

秘笈心法

由于在页面初始化时会自动执行<body>标签中的内容，因此可以将<script>标签的内容直接写到<body>标签体内，这样也可以实现本实例的效果。

实例 064 控制表格指定外边框不显示

光盘位置：光盘\MR\03\064

初级

实用指数：★★★☆

实例说明

本实例将介绍如何应用 CSS 样式实现表格外边框不显示。运行本实例，在页面中将看到两个表格，其中一个表格显示外边框，而另一个表格不显示外边框，实例运行效果如图 3.13 所示。

商品ID	商品名称	单位	单价
0001	苹果	箱	30
0002	XX香烟	条	50
0003	XX方便面	箱	20

表(1)

商品ID	商品名称	规格	单价
0001	苹果	箱	30
0002	XX香烟	条	50
0003	XX方便面	箱	20

表(2)

图 3.13　控制表格指定外边框不显示

关键技术

本实例主要应用 CSS 样式的边框属性中的 border-width 来实现，border-width 表示边框的粗细，单位为像素，当此属性值为 0 时，表示不显示边框。

设计过程

（1）新建 index.htm 网页，在该网页的<style>标签中编写带有外边框表格的 CSS 样式，关键代码如下：

```
.table1{
        border-left-style:solid;
        border-bottom-style:solid;
        border-right-style:solid;
        border-top-style:solid;
        border-width: 2px;
        border-color:green;
}
```

（2）编写不带边框的表格样式，并设置表格的背景颜色，关键代码如下：

```
.table2{
        border-width: 0px;
        background-color: pink;
}
```

（3）在页面中添加两个表格，分别使用不同的 CSS 样式，关键代码如下：

```
<body>
<table id="tb1" class="table1">
  <tr height="30">
        <td>员工编号</td><td>员工姓名</td><td>员工性别</td><td>员工年龄</td>
  </tr>
  <tr>
         <td >001</td><td >张三</td><td >男</td><td >35</td>
  </tr>
  <tr>
         <td>002</td><td>李四</td><td>男</td><td>28</td>
  </tr>
   <tr>
         <td>003</td><td>王二</td><td>男</td><td>27</td>
  </tr>
</table>
<table id="tb2" class="table2">
  <tr height="30">
       <td>员工编号</td><td>员工姓名</td><td>员工性别</td><td>员工年龄</td>
  </tr>
  <tr>
       <td >001</td><td >张三</td><td >男</td><td >35</td>
  </tr>
  <tr>
        <td>002</td><td>李四</td><td>男</td><td>28</td>
  </tr>
   <tr>
         <td>003</td><td>王二</td><td>男</td><td>27</td>
  </tr>
</table>
</body>
```

秘笈心法

CSS 样式中的边框属性不是表格所特有的，在 HTML 元素中有些元素也具有边框属性，如图片<img>、文本框<input>、文本域<textarea>等。因此在设计网页时，可以通过边框属性来灵活应用在不同的 HTML 元素中。

实例 065　背景颜色渐变的表格

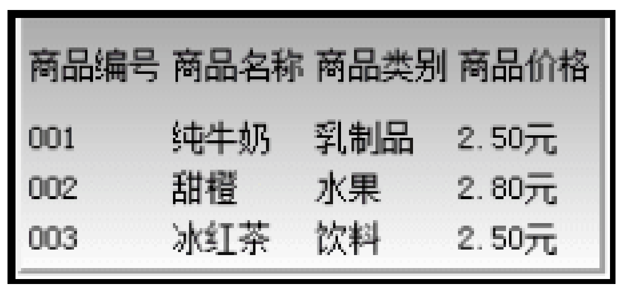
光盘位置：光盘\MR\03\065

初级

实用指数：★★★☆

实例说明

在制作网站时，网站中加入的表格可以制作成渐变的效果。本实例将通过 CSS 样式来设置表格的渐变效果，运行效果如图 3.14 所示。



商品编号	商品名称	商品类别	商品价格
001	纯牛奶	乳制品	2.50元
002	甜橙	水果	2.80元
003	冰红茶	饮料	2.50元

图 3.14　背景颜色渐变的表格

关键技术

本实例主要应用 CSS 样式的 progid 滤镜实现，其语法如下：

```
filter:progid:dximageTramsform.Microsoft.Gradient(enabled=bEnabled,startColorStr=colorStr,endColorStr=colorStr,)
```

参数说明

❶ enabled：设置或检索滤镜是否激活。默认值为 true，为 false 时表示禁止滤镜。

❷ startColorStr：设置或检索色彩渐变的开始颜色和透明度，默认值为#000000FF（不透明的蓝色）。

❸ endColorStr：设置或检索色彩渐变的结束颜色和透明度，默认值为#000000FF（不透明的黑色）。

设计过程

（1）新建 index.htm 网页，在该网页中添加一个表格，关键代码如下：

```
<table   border="1">
  <tr height="30">
    <td>商品编号</td><td>商品名称</td><td>商品类别</td><td>商品价格</td>
  </tr>
  <tr>
    <td >001</td><td >纯牛奶</td><td >乳制品</td><td >2.50 元</td>
  </tr>
  <tr>
    <td>002</td><td>甜橙</td><td>水果</td><td>2.80 元</td>
  </tr>
   <tr>
    <td>003</td><td>冰红茶</td><td>饮料</td><td>2.50 元</td>
  </tr>
</table>
```

（2）在页面的<style>标签中编写 CSS 样式，使表格背景渐变，关键代码如下：

```
<style type="text/css">
table{
        font-size:12px;
        border-width:2px;
        filter:progid:dximageTransform.Microsoft.Gradient(fradientType=1,
        startColorStr=#46BEFF,endColorStr=#FFFFFF);
}
td{
        border-width:0px;
}
</style>
```

秘笈心法

为了提高开发效率，可以将 progid 滤镜的样式代码保存到.css 样式文件中，在不同网页中应用此样式时直接通过<link>标签引用 CSS 文件即可，并不需要死记硬背这些滤镜属性。

实例 066	表格隔行变色 光盘位置：光盘\MR\03\066	初级 实用指数：★★★☆

实例说明

在很多网站中，表格的隔行变色的效果应用得非常多。本实例将通过 CSS 样式来控制表格的隔行变色，运行效果如图 3.15 所示。

图书编号	图书名称	作者	图书价格
001	《JavaWeb编程宝典》	明日科技	98.00元
002	《Struts2深入详解》	孙鑫	79.00元
003	《Tomcat与JavaWeb技术详解》	孙卫琴	79.50元
004	《Java编程思想》第4版	Bruce Eckel	108.00元

图 3.15　隔行变色的表格

关键技术

本实例主要通过在 CSS 样式中使用 JavaScript 中的 split()方法来改变表格的背景颜色 backgound-color 属性值。split()方法的语法如下：

```
stringObj.split(str)[Index]
```

参数说明

❶ stringObj：要被分解的 String 对象或文字。

❷ str：用来定义以什么字符分解。

❸ Index：表示下一个匹配从该索引处开始。

功能：split()方法通过字符 str 将 strObj 字符串分隔成一个字符串数组。

在实现隔行变色的样式时，还需要使用 expression 将 CSS 属性和 JavaScript 关联起来，而且需要在 split()

方法的 index 索引中使用表格的行索引对象 rowIndex。

设计过程

（1）新建 index.htm 网页，在该网页中的<style>标签中编写隔行变色的 CSS 样式，关键代码如下：

```
<style type="text/css">
.changeTr tr{
        font-size:12px;
        border-width:0px;
        background-color: expression("#DEF1FE,#FFFFFF".split(",")[rowIndex%2]);
}
td{
        border-width:0px;
}
</style>
```

（2）在页面中添加表格，并将表格的 class 属性值设置为 CSS 样式名 changeColor，关键代码如下：

```
<table   border="0" class="changeTr">
```

秘笈心法

JavaScript 中的 split()方法在实际开发中经常使用，如从一个日期格式的字符串（yyyy-mm-dd）中获取年、月、日的值时，就需要使用到该方法来实现。

实例 067　表格隔列变色

光盘位置：光盘\MR\03\067

初级

实用指数：★★★☆

实例说明

在访问网站时，有些显示数据的表格样式会有隔列变色的效果。本实例将通过 CSS 样式来设置此变色效果，运行效果如图 3.16 所示。

图书编号	图书名称	作者	图书价格
001	《JavaWeb编程宝典》	明日科技	98.00元
002	《Struts2深入详解》	孙鑫	79.00元
003	《Tomcat与JavaWeb技术详解》	孙卫琴	79.50元
004	《Java编程思想》第4版	Bruce Eckel	108.00元

图 3.16　隔列变色的表格

关键技术

通过 CSS 设置表格的隔列变色与隔行变色类似，同样需要使用 JavaScript 的 split()函数。与表格隔行变色的实现不同的是，隔列变色需要在 CSS 样式中使用表格的单元格索引对象 cellIndex，然后通过 split()方法，实现每隔一个单元格，就将单元格背景颜色改变。

设计过程

（1）新建 index.htm 网页，在该网页中的<style>标签中编写隔列变色的 CSS 样式，关键代码如下：

```
<style type="text/css">
.changeTd td{
        font-size:12px;
        border-width:0px;
        background-color: expression("#DEF1FE,orange".split(",")[cellIndex%2]);
}
td{
        border-width:0px;
}
</style>
```

（2）在网页中添加表格，并设置表格的 class 属性值为 CSS 样式名 changeTd，关键代码如下：

```
<table border="0" class="changeTd" >
```

秘笈心法

在实际应用中，经常需要在 JavaScript 中应用表格对象的属性来动态制作表格，如表格行索引 rowIndex 和

单元格索引 cellIndex。JavaScript 中的表格对象具有很多属性和方法，这些属性和方法会在后面的章节中进行详细讲解。

实例 068 鼠标经过表格时，显示提示信息

光盘位置：光盘\MR\03\068

初级

实用指数：★★★☆

实例说明

在浏览网站信息时，当鼠标经过表格的某个单元格时，会显示出相关的提示信息。运行本实例，当鼠标经过表格中的某个“图书名称”时，将显示提示信息，运行效果如图 3.17 所示。

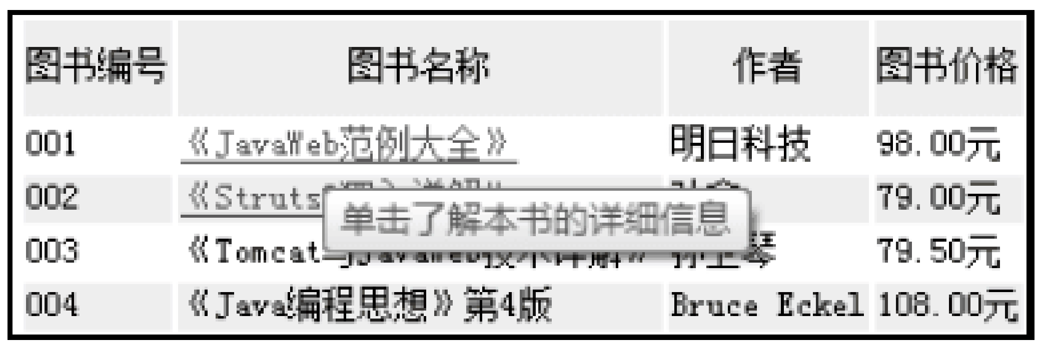


图 3.17　显示提示信息的表格

关键技术

实现本实例非常简单，主要通过设置表格<td>中的 title 属性来实现，title 属性不是表格特有的属性，HTML 中的绝大多数标签都具有该属性。

设计过程

新建 index.htm 网页，在该网页中添加一个表格，并在“图书名称”的单元格中设置 title 属性，关键代码如下：

```
<table   border="0" class="changeTd" >
  <tr height="30" >
    <td align="center" >图书编号</td><td align="center">图书名称</td>
    <td align="center">作者</td><td align="center">图书价格</td>
  </tr>
  <tr>
<td >001</td>
<td title="单击了解本书的详细信息">
            <a href="#">《JavaWeb 范例大全》</a>
    </td>
    <td >明日科技</td><td >98.00 元</td>
  </tr>
  <tr>
    <td>002</td>
<td title="单击了解本书的详细信息">
<a href="#">《Struts2 深入详解》</a>
</td>
    <td>孙鑫</td><td>79.00 元</td>
  </tr>
   <tr>
    <td>003</td><td>《Tomcat 与 JavaWeb 技术详解》</td><td>孙卫琴</td><td>79.50 元</td>
  </tr>
   <tr>
    <td>004</td><td>《Java 编程思想》第 4 版</td><td>Bruce Eckel</td><td>108.00 元</td>
  </tr>
</table>
```

秘笈心法

根据本实例的实现，读者可以设置当鼠标经过图片时，显示图片的详细信息。在网页中添加图片使用的是<img>标签，只要设置相应<img>标签的 title 属性即可。

3.3　鼠 标 样 式

在设计网页时，为了使网页看上去更加美观，并且能够吸引浏览者，通常会设计鼠标指针的特效，针对不

同的操作，可以显示不同样式的鼠标指针形状。下面将通过几个例子来介绍如何改变鼠标的样式。

实例 069　显示自定义的鼠标形状

光盘位置：光盘\MR\03\069

初级

实用指数：★★★★

实例说明

运行本实例，如图 3.18 所示，当鼠标指针经过按钮、表格以及超链接时，鼠标指针会变为不同的形状，当移出时会恢复初始状态。



图 3.18　显示自定义的鼠标形状

关键技术

本实例主要通过在 CSS 样式中，设置 cursor 的属性值来实现不同的鼠标指针形状，cursor 属性有多种属性值，每个值都代表一个不同形状的鼠标形状，具体的属性及说明如表 3.7 所示。

表 3.7　cursor 属性及说明

cursor 属性	说　明
hand	手形指针
crosshair	交叉十字形指针
text	文本选择符号
wait	沙漏形状的等待指针
default	默认形状的指针
help	带问号形状的指针
move	移动的箭头
e-resize	向右的箭头 →
ne-resize	向右上的箭头 ↗
n-resize	向上的箭头 ↑
nw-resize	向左上的箭头 ↖
w-resize	向左的箭头 ←
sw-resize	向左下的箭头 ↙
s-resize	向下的箭头 ↓
se-resize	向右下的箭头 ↘

设计过程

（1）新建 index.htm 网页，在该页的<style>标签中分别编写表格、超链接、单元格以及按钮的 CSS 样式，关键代码如下：

```
<style type="text/css">
table{
        font-size:12px;
```

```
        background-color:activecaption;
        cursor: crosshair;
}
a{
        cursor:help;
}
td{
        cursor:text;
}
.btn{
        font-size:12px;
        cursor:hand;
}
</style>
```

（2）在页面中添加表格、超链接和按钮等，关键代码如下：

```
<body >
<table   border="0" >
  <tr height="30" >
    <td align="center" >图书编号</td><td align="center">图书名称</td>
    <td align="center">作者</td><td align="center">图书价格</td>
  </tr>
  <tr>
    <td >001</td>
    <td>
            <a href="#">《JavaWeb 范例大全》</a>
    </td>
    <td >明日科技</td><td >98.00 元</td>
  </tr>
  <tr>
    <td>002</td>
    <td><a href="#"> 《Struts2 深入详解》</a></td>
    <td>孙鑫</td><td>79.00 元</td>
  </tr>
</table>
<input type="button" value="提  交" class="btn"/>
</body>
```

秘笈心法

鼠标指针 cursor 属性经常被用在按钮和图片上，当鼠标指针经过按钮时，通常会设置 cursor 属性值为 hand，将指针形状变成手形，这样更符合实际需求。相对而言，cursor 属性的其他属性值并不常用。

实例 070	动画光标 光盘位置：光盘\MR\03\070	初级 实用指数：★★★★

实例说明

在浏览一些个性网站时，如博客、QQ 空间，其网站中的光标常常采用动画光标，这样可以突出网站的个性风格。本实例将介绍如何在网页中加入动画光标，运行本实例，当鼠标在页面工作区内时，将光标改变成指定的动画效果。

关键技术

本实例主要通过在 CSS 样式中设置 cursor 的 url 属性来实现，url 表示一个动态光标的文件路径，一般动态光标是一个.ani 文件。

设计过程

新建 index.htm 网页，在该页的<style>标签内编写动态光标的 CSS 样式，关键代码如下：

```
<style type="text/css">
body{
        cursor:url('g.ani');
}
</style>
```

秘笈心法

目前，有很多种制作动画光标的工具，这些工具在网站上都可以找到，使用这些工具可以制作出自己喜爱的各种样式的光标。

3.4　文字及列表样式

在制作网页时，经常需要将网页中的文字以及列表设置成不同的样式。下面将通过几个例子来介绍如何使用 CSS 样式设置文字和列表的样式。

实例 071	应用删除线样式标记商品特价 光盘位置：光盘\MR\03\071	初级 实用指数：★★★★

实例说明

在浏览一些网上商店时，商品列表中的有些商品是显示有折扣的，一般会显示商品原价和打折之后的价格，显示原价的文字会带有删除线，本实例的运行效果如图 3.19 所示。

图书编号	图书名称	作者	图书原价	现价
001	《JavaWeb范例大全》	明日科技	~~98.00元~~	79.00元
002	《Struts2深入详解》	孙鑫	~~79.00元~~	59.00元
003	《Tomcat与JavaWeb技术详解》	孙卫琴	~~79.50元~~	69.00元
004	《Java编程思想》第4版	Bruce Eckel	~~108.00元~~	89.00元

图 3.19　带删除线的商品价格

关键技术

文字加删除线，主要使用的是 CSS 样式中的 text-decoration 属性。该属性有 5 个可选值，即 underline（文字加下划线）、overline（文字加上划线）、line-through（文字加删除线）、blink（闪烁文字，只有 Netscape 浏览器支持）和 none（默认值）。

设计过程

（1）新建 index.htm 页面，在该页面的<style>标签中编写页面的 CSS 样式，关键代码如下：

```
<style type="text/css">
font{
        font-size:12px;
        color: red;
        text-decoration: line-through;
}
table{
        font-size:12px;
        background-color:threedface;
        border: 1px solid;
}
</style>
```

（2）在页面中，将商品原价文字写在<font>标签之间，关键代码如下：

```
<td>
        <font>98.00 元</font>
</td>
```

秘笈心法

text-decoration 属性的 line-through（文字加删除线）属性值经常应用在显示商品价格的文字上，所以有必要记住这个属性的用法。

实例 072 在文字上方标注说明标记

光盘位置：光盘\MR\03\072

初级

实用指数：★★★★

实例说明

在浏览一些信息网站时，网站中发表的一些具有专业技术的文章可能会有很多词汇不容易理解，这就需要对这些词汇进行解释，而为了更直观地了解词汇的含义，可以在文字上方添加标记说明。本实例将介绍如何在文字的上方加入标注说明，运行效果如图 3.20 所示。

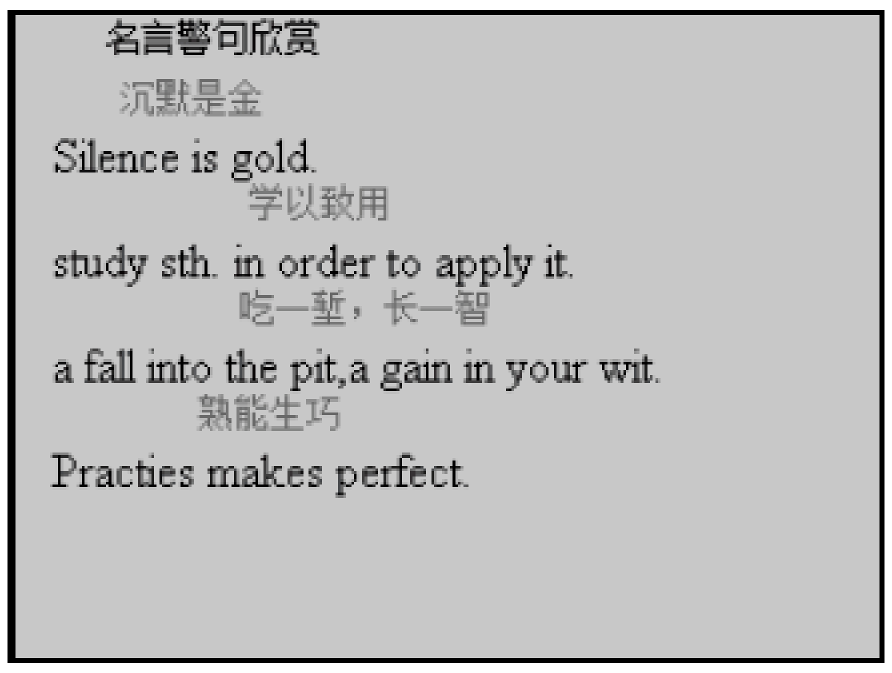


图 3.20　在文字上方添加标记说明

关键技术

实现在文字上方添加标注说明，主要是在页面中使用<rt>和<ruby>标签，<rt>标签包含在<ruby>标签内，然后将文字写在<rt>标签内，此时<rt>标签内的文字会显示在正常文字的上方，使用方法如下：

```
<ruby>
        正常显示的文字
<rt>文字上方的说明标记<rt>
<ruby>
```

设计过程

新建 index.htm 页面，在该页面的<body>标签中添加文字的标记说明标签，关键代码如下：

```
<ruby>
        Silence is gold.
        <rt><font>沉默是金</font></rt><br>
</ruby>
<ruby>
        study sth. in order to apply it.
        <rt><font>学以致用</font></rt><br>
</ruby>
<ruby>
        a fall into the pit,a gain in your wit.
        <rt><font>吃一堑，长一智</font></rt><br>
</ruby>
<ruby>
        Practies makes perfect.
        <rt><font>熟能生巧</font></rt>
</ruby>
```

秘笈心法

<rt>和<ruby>标签在实际应用中并不常见，所以了解一下它们的用处即可。

实例 073 改变首行文字的样式

光盘位置：光盘\MR\03\073

初级

实用指数：★★★★

实例说明

在浏览网站阅读一些文章时，通常会见到在首行文字上有一些特殊的样式。本实例将通过 CSS 样式来改变首行文字的样式，运行效果如图 3.21 所示。

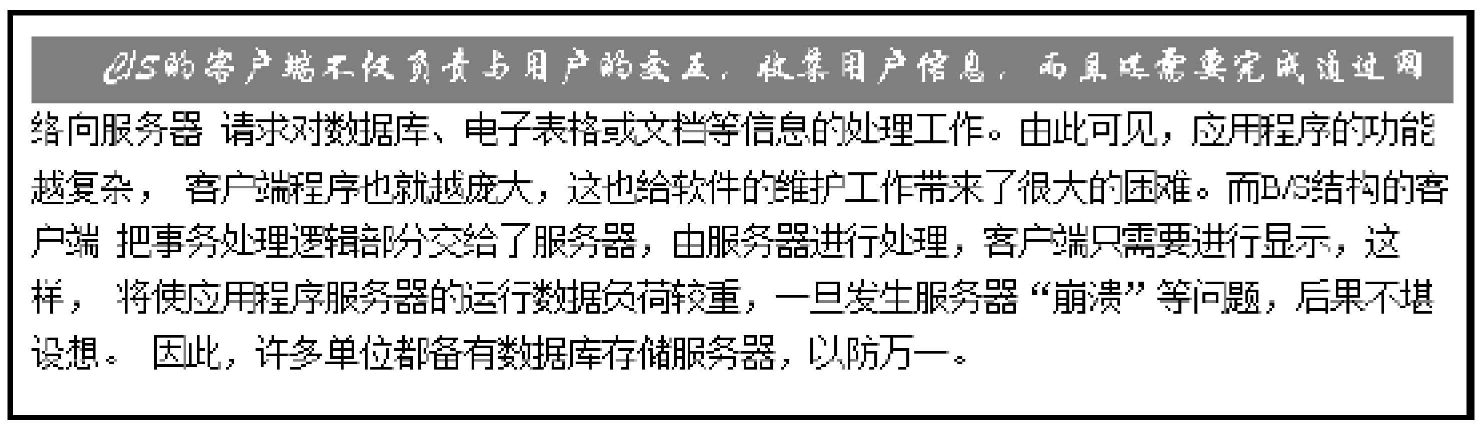


图 3.21　改变首行文字的样式

关键技术

本实例通过在 CSS 样式中设置段落<p>标签首行对象 first-line 的属性来改变首行文字的样式。first-line 表示段落的首行，其 CSS 样式的语法格式如下：

```
p:first-line{ font-size:12px; background :pink; color:#FFFFFF;...};
```

设计过程

（1）新建 index.htm 页面，在<style>标签中编写<p>标签的 CSS 样式，关键代码如下：

```
<style>
p {
	line-height: 1.6;
	font-size: 14px;
	font-family:  华文细黑;
}
p :FIRST-LINE {
	color:#FFFFFF;
	background:#FF6600;
	font-size: 16px;
	font-family:  华文行楷;
}
</style>
```

（2）在页面中添加段落<p>标签，并添加一段文字内容，关键代码如下：

```
<body>
<table width="580" align="center" >
	<tr>
		<td>
			<p>
				此处添加文字内容
			</p>
		</td>
	</tr>
</table>
</body>
```

秘笈心法

在 CSS 样式中，除了 first-line 属性，段落标签<p>还包括 first-letter 属性，表示段落的首个文字的属性。

实例 074　使文字具有下划线效果

光盘位置：光盘\MR\03\074

初级

实用指数：★★★★

实例说明

在浏览网站信息时，为了更突出显示一些重要信息，经常会在这些重要信息的文字处加上下划线。本实例将通过 CSS 样式来设置文字的下划线效果，实例运行效果如图 3.22 所示。

关键技术

本实例主要通过设置 CSS 样式的 text-decoration 属性来实现，该属性包含 5 个可选值，即 underline（文字加下划线）、overline（文字加上划线）、line-through（文字加删除线）、blink（闪烁文字，只有 Netscape 浏览器

支持）和 none（默认值）。

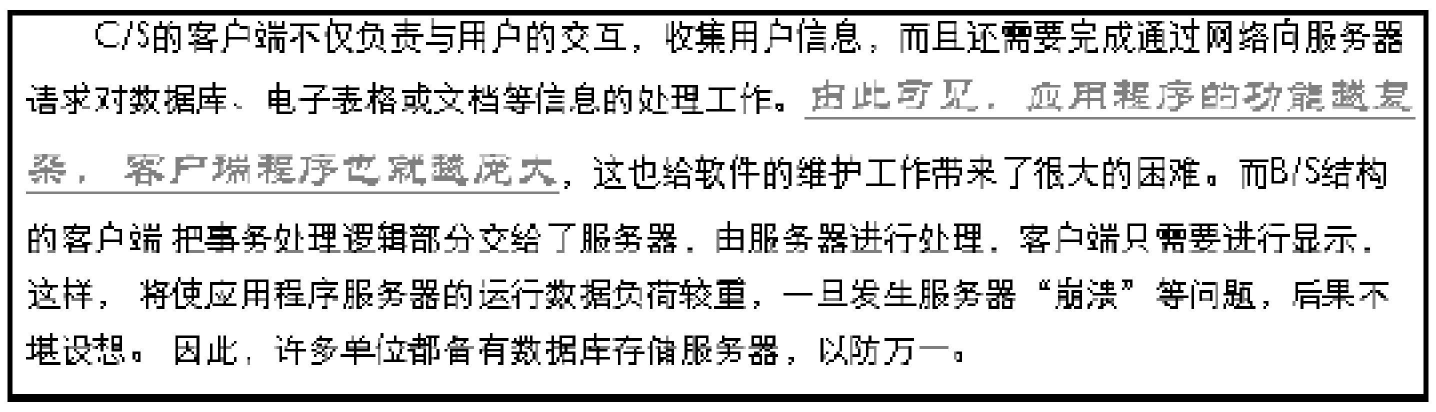


图 3.22　文字带下划线的样式

设计过程

（1）新建 index.htm 页面，在<style>标签中编写下划线文字的 CSS 样式，关键代码如下：

```
<style>
font{
        font-size:18px;
        text-decoration: underline;
        color: gray;
        font-family:  华文隶书;
}
p{
        line-height: 1.6;
        font-size: 14px;
        font-family:  华文细黑;

}
</style>
```

（2）在页面中添加<font>标签，在该标签内输入测试的文字内容，关键代码如下：

```
<font>
        由此可见，应用程序的功能越复杂，客户端程序也就越庞大
</font>
```

秘笈心法

text-decoration 属性的 underline（文字加下划线）属性值，经常应用在一些网站的文章中的某一段重要文字或某句话上，所以有必要记住这个属性的用法。

实例 075　指定图标的列表项

光盘位置：光盘\MR\03\075

初级

实用指数：★★★★

实例说明

在默认情况下，列表标记只能是符号，为了使页面更加美观，可以使用 CSS 样式将列表标记设置成图标，运行效果如图 3.23 所示。



图 3.23　带有图标的列表项

关键技术

在网页中的列表项以<ul>标签表示，所以可以在 CSS 样式中通过设置<ul>的属性来实现列表的样式。本实例通过在 CSS 中设置列表的 list-style-image 属性来改变列表的标记为图标，语法格式如下：

```
ul { list-style-image: url(ico.gif); ...}
```

其中 url 中的内容是图标文件的路径。

设计过程

（1）新建 index.htm 页面，在<style>标签中编写列表的 CSS 样式，关键代码如下：

```
<style>
body{
```

```
        font-size:12px;
}
ul {
        list-style-image: url(star.gif);
        color: gray;
        font-family: 华文细黑;
}
</style>
```

（2）在页面中添加列表标签<ul>，关键代码如下：

```
<ul>
<li>request：请求对象</li>
<li>response：响应对象</li>
<li>session：会话对象</li>
<li>application：应用程序对象</li>
<li>out：输出对象</li>
<li>...</li>
</ul>
```

秘笈心法

根据本实例，读者可以实现在电子商城中的商品类别列表中加入图标。

3.5　文 字 特 效

在浏览网页时，经常会到看到一些显示特效的文字，这些特效文字多数都是利用 CSS 样式来实现的。下面将通过几个例子来介绍如何使用 CSS 样式来设置文字的特效。

实例 076	文字的发光效果 光盘位置：光盘\MR\03\076	初级 实用指数：★★★★

实例说明

在设计网站时，为了体现网站的个性，可以设计网站导航或其他内容的文字发光效果。本实例将通过 CSS 样式来设置文字的发光特效，运行结果如图 3.24 所示。



图 3.24　文字的发光效果

关键技术

本实例主要通过 CSS 样式的 glow 滤镜属性实现，其语法格式如下：

```
{filter: glow(color=color_value,Strength=value)}
```

参数说明

❶ color：表示边缘光晕的颜色。

❷ Strength：表示边缘光晕的强度大小，范围在 1～255 之间。

设计过程

（1）新建 index.htm 页面，编写文字发光效果的 CSS 样式，关键代码如下：

```
<style>
.tb{
        font-size:40px;
        filter:glow(color=skyblue,direction=2);
        font-family: 华文行楷;
}
</style>
```

（2）在页面中添加表格，并使用以上定义的 CSS 样式，关键代码如下：

```
<table class="tb">
<tr>
        <td>
                明日科技论坛
        </td>
</tr>
</table>
```

秘笈心法

为了提高开发效率，可以将 glow 滤镜的样式代码写到.css 样式文件中，在需要加入特效的文字中，直接通过<link>标签引用 CSS 文字滤镜样式文件即可，避免每次都编写该文字特效的代码。

实例 077 文字的阴影效果

光盘位置：光盘\MR\03\077

初级

实用指数：★★★★

实例说明

在设计网站时，为了体现网站的个性，可以设计网站内容的文字阴影效果。本实例将通过 CSS 样式来设置文字的阴影效果，运行结果如图 3.25 所示。



图 3.25　文字的阴影效果

关键技术

本实例主要通过 CSS 样式的 dropshadow 滤镜属性实现，其滤镜语法格式如下：

```
{filter: dropshadow(color=color_value,offx=value,offy=value,positive=value)}
```

参数说明

❶ color：阴影的颜色。

❷ offx：x 轴方向阴影的偏移量。

❸ offy：y 轴方向阴影的偏移量。

❹ positive：取值为 true 或 false，true 表示为不透明像素建立可见的阴影，false 表示为透明像素建立可见的阴影。

设计过程

（1）新建 index.htm 页面，编写文字阴影效果的 CSS 样式，关键代码如下：

```
<style>
.tb{
        font-size:40px;
        filter:dropshadow(color=green,offx=3,offy=3);
        font-family: 华文隶书;
}
</style>
```

（2）在页面中添加表格，并使用以上定义的 CSS 样式，关键代码如下：

```
<table class="tb">
<tr>
        <td>
                明日科技论坛
        </td>
</tr>
</table>
```

秘笈心法

在 IE 10 及以上版本中，如果要实现文字的阴影效果，可以使用 CSS 3 提供的 text-shadow 属性来实现。

实例 078　文字的渐变阴影效果

光盘位置：光盘\MR\03\078

初级

实用指数：★★★★

实例说明

在设计网站时，为了体现网站的个性，可以设计网站内容的文字渐变阴影效果。本实例将通过 CSS 样式来设置文字的渐变阴影效果，运行结果如图 3.26 所示。



图 3.26　文字的渐变阴影效果

关键技术

本实例主要通过 CSS 样式的 shadow 滤镜属性实现，其语法格式如下：

```
{filter: shadow(color=color_value,direction=value)}
```

参数说明

❶ color：阴影的颜色。

❷ direction：设定渐变阴影的方向，渐变阴影是按顺时针方向进行的，0° 表示垂直向上，然后每 45° 为一个单位，默认值为 270° 。

设计过程

（1）新建 index.htm 页面，编写文字渐变阴影效果的 CSS 样式，关键代码如下：

```
<style>
.tb{
	font-size:40px;
	filter:shadow(color=skyblue,direction=135);
	font-family: 黑体;
}
</style>
```

（2）在页面中添加表格并应用以上定义的 CSS 样式，关键代码如下：

```
<table class="tb">
<tr>
	<td>
		明日科技论坛
	</td>
</tr>
</table>
```

秘笈心法

为了提高开发效率，可以将 shadow 滤镜的样式代码写到.css 样式文件中，在需要加入特效的文字中，直接通过<link>标签引用 CSS 文字滤镜样式文件即可，避免每次都编写该文字特效的代码。

实例 079　文字的图案填充效果

光盘位置：光盘\MR\03\079

高级

实用指数：★★★★

实例说明

在设计网站时，为了体现网站的个性，可以设计网站 Logo 文字的图案填充效果。本实例将通过 CSS 样式来设置文字的图案填充效果，运行结果如图 3.27 所示。



图 3.27　文字的图案填充效果

关键技术

本实例主要通过 CSS 样式的 dropshadow 滤镜和 chroma 滤镜实现。dropshadow 滤镜的用法请参考实例 077，

chroma 滤镜的语法格式如下：

```
{filter: chroma(color=color_value)}
```

参数说明

color：透明的颜色，如 color 的值为#FFCC00，则滤镜作用范围内的所有为#FFCC00 的颜色都变为透明，包括图片的像素。

设计过程

（1）新建 index.htm 页面，在页面中添加<div>标签，并设置<div>的背景图片，关键代码如下：

```
<div style="background-image:url(bg.gif);width:100%;">
```

（2）在步骤（1）添加的<div>标签中，再添加一个<div>标签，然后添加文字，并设置 div 中的文字滤镜样式，关键代码如下：

```
<div style="FILTER: chroma( color=#CCCCCC) dropShadow(Color=#777777,offX=-1,offY=-1,positive=2);
        color: #CCCCCC; background-color: #FFFFFF;width: 100%; font: bold 50pt  黑体; ">
        明日科技论坛
</div>
```

秘笈心法

在应用 chroma 滤镜实现文字的填充效果时，经常需要再应用 dropshadow 滤镜为文字设置阴影效果，这样可以让文字的填充效果更加清晰、美观。

实例 080	文字的探照灯效果 光盘位置：光盘\MR\03\080	高级 实用指数：★★★★

实例说明

在设计网站时，为了体现网站的个性，可以设计网站 Logo 文字的探照灯效果。本实例将通过 CSS 样式来设置文字的探照灯效果。当鼠标在文字上移动时，文字上方会有类似于灯光的照明效果，运行结果如图 3.28 所示。

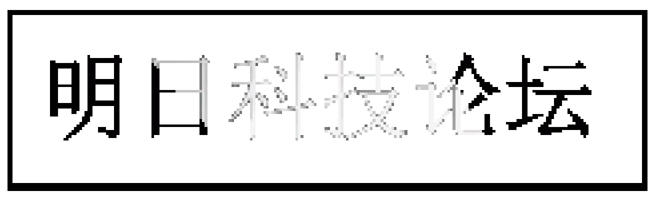


图 3.28　文字的探照灯效果

关键技术

本实例主要通过 CSS 样式的 light 滤镜实现，其语法格式如下：

```
{filter: light}
```

在 JavaScript 中，可以使用 document 对象来获得元素的 light 滤镜对象，然后通过调用 light 滤镜对象中的方法来实现文字的发光效果，light 滤镜中包含的方法及说明如表 3.8 所示。

表 3.8　light 滤镜的方法及说明

方　　法	说　　明
addAmbient()	加入包围的光源
addCone()	加入锥形光源
addPoint()	加入点光源
changcolor()	改变光的颜色
changstrength()	改变光源的强度
clear()	清除所有光源
moveLight()	移动光源

在 JavaScript 方法中，调用 addAmbient()方法的语法如下：

```
document.all.Obj.filters.light.addAmbient(R,G,B,strength)
```

其中 Obj 表示 HTML 标签的 id 属性值。参数 R、G、B 表示红、绿、蓝的颜色分量，strength 表示光投射的数量。

调用 addCone()方法的语法如下：

```
document.all.Obj.filters.light.addCone(x1,y1,z1,x2,y2,z2,R,G,B,strength,spread)
```

参数 x1、y1、z1 表示光源的位置，x2、y2、z2 表示目标点的位置，R、G、B 表示红、绿、蓝的颜色分量，strength 表示光投射的数量，spread 表示光源的范围。

调用 moveLight()方法的语法如下：

```
document.all.Obj.filter.light.moveLight(lightNumber,x,y,z,fAbsolute)
```

参数 lightNumber 表示光源的数字，x、y、z 表示光源的三维坐标，fAbsolute 表示移动是相对的还是绝对的。

设计过程

（1）新建 index.htm 页面，编写<div>标签的 CSS 样式，关键代码如下：

```
<style>
.mydiv{
	color:white;
	filter:light;
	height:300;
	font-size:35px;
	left: 10px;
	position: relative;
	top:10px;
	width: 400px;
}
</style>
```

（2）在页面中添加一个<div>标签，并且应用以上定义的 CSS 样式，关键代码如下：

```
<div id="div1" class="mydiv">
	明日科技论坛
</div>
```

（3）在 JavaScript 中，调用 light 滤镜对象的方法，设置光源，关键代码如下：

```
<script type="text/javascript">
/**初始化方法 */
function init(){
	document.all.div1.filters.light.addAmbient(10,10,40,30);
	document.all.div1.filters.light.addCone(400,400,120,160,100,255,255,80,120,5);
}
/**鼠标经过时调用的方法 */
function move(){
	document.all.div1.filters.light.moveLight(1,window.event.x-20,window.event.y,0,1);
}
</script>
```

（4）在页面的 onload 事件中调用 init()方法，onMousemove 事件中调用 move()方法，关键代码如下：

```
<body onload="init()" onmousemove="move()">
```

秘笈心法

本实例中应用的 light 滤镜的方法在实际应用中并不常见，因此只要了解一下该滤镜的用法即可，无须死记硬背。

实例 081	文字的闪烁效果 光盘位置：光盘\MR\03\081	高级 实用指数：★★★★

实例说明

本实例将通过 CSS 样式来设置文字的闪烁效果。当页面加载后，文字开始闪烁，当鼠标移动到文字上时，文字停止闪烁；鼠标移出文字时，文字继续闪烁，运行结果如图 3.29 所示。



图 3.29　文字的闪烁效果

关键技术

本实例主要通过 CSS 样式的 glow 滤镜实现，glow 滤镜的用法请参考实例 076。实现文字的闪烁效果主要

是使用 JavaScript 函数来控制 glow 滤镜的 enabled 属性，enabled 属性表示 glow 滤镜是否被禁用，当 enabled 的属性值为 true 时则激活 glow 滤镜，否则禁用 glow 滤镜。

设计过程

（1）新建 index.htm 页面，编写文字发光的 CSS 样式和层的相对位置样式，关键代码如下：

```
<style type="text/css">
.glow1 {
 position:absolute; width:296px; height:42px; z-index:1;
 filter:glow(color=#FF0000,strengh=2)
}
.inside   {
 position:relative;top:10px;
}
</style>
```

（2）在页面中添加层并且在层中添加文字，关键代码如下：

```
<div id="div1" align="center" class="glow1" onmouseover="stopflash(this)" onmouseout="init()">
        <div class="inside">文字的闪烁效果</div>
</div>
```

（3）编写实现文字闪烁效果的 JavaScript 方法，关键代码如下：

```
<script language="javascript">
/**页面加载后调用的方法 */
function init(){
        makeflash(div1);
}
/**光晕开始闪烁
*@param obj:div 标签对象
*/
function makeflash(obj){
        //每隔一秒，禁用 glow 滤镜一次，1000 表示 1000 毫秒，即 1 秒
        obj.flashTimer=setInterval("div1.filters.glow.enabled= !div1.filters.glow.enabled",1000);
}
/**光晕停止闪烁 */
function stopflash(obj){
        clearInterval(obj.flashTimer)
}
</script>
```

秘笈心法

本实例应用到了 JavaScript 中 window 对象的 setInterval()方法，该方法会根据指定的时间间隔来执行 JavaScript 代码，该方法的语法结构如下：

```
var timer = setInterval(func,timeValue);
```

参数 func 表示将要执行的 JavaScript 代码或 JavaScript 自定义的方法，timeValue 表示时间间隔，以毫秒为单位。

实例 082	文字的空心效果 光盘位置：光盘\MR\03\082	高级 实用指数：★★★★

实例说明

在设计网站时，为了体现网站的个性，可以设计网站内容文字的空心效果。本实例将通过 CSS 样式来设置文字的空心效果，运行效果如图 3.30 所示。

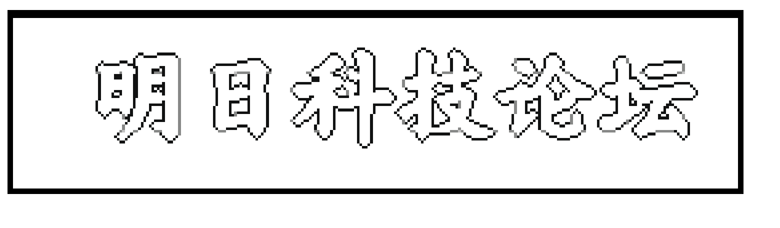


图 3.30　文字的空心效果

关键技术

本实例主要通过 CSS 样式的 glow 滤镜、mask 滤镜和 chroma 滤镜实现。glow 和 chroma 滤镜的用法在前面实例中已经介绍过，mask 滤镜的语法如下：

```
{filter: mask(color= color_value)}
```

参数说明

❶ mask：主要用于文字遮罩。

❷ color：表示遮罩所用的颜色。

设计过程

（1）新建 index.htm 页面，编写文字空心效果的 CSS 样式，关键代码如下：

```
<style type="text/css">
.tb1{
        font-size: 40px;
        font-family:楷体;
        FILTER: glow(strength=1)mask(color=blue)chroma(color=blue)
}
</style>
```

（2）在页面中添加表格，并应用以上定义的 CSS 样式，关键代码如下：

```
<table align="center"   height="60" border="0" class="tb1">
   <tr>
      <td height="60" align="center">明日科技论坛</td>
   </tr>
</table>
```

秘笈心法

为了提高开发效率，可以将空心效果的文字样式代码写到.css 样式文件中，在需要加入特效的文字中，直接通过<link>标签引用 CSS 文字滤镜样式文件即可，避免每次都编写该文字特效代码。

实例 083	文字的浮雕效果 光盘位置：光盘\MR\03\083	高级 实用指数：★★★★

实例说明

在设计网站时，为了体现网站的个性，可以设计网站导航文字的浮雕效果。本实例将通过 CSS 样式来设置文字的浮雕效果，运行效果如图 3.31 所示。



图 3.31　文字的浮雕效果

关键技术

本实例主要通过 CSS 样式的 mask 滤镜、shadow 滤镜、dropshadow 滤镜和 chroma 滤镜实现。首先使用 mask 滤镜将可见像素遮蔽，将看不见的像素以指定颜色显示，然后使用 shadow 滤镜使文字具有渐变阴影效果，之后用 dropshadow 滤镜使文字具有阴影效果，最后使用 chroma 滤镜将指定的颜色进行透明处理。

设计过程

（1）新建 index.htm 页面，编写文字浮雕效果的 CSS 样式，关键代码如下：

```
<style type="text/css">
.tb1 {
font-size:60px;
font-family:隶书;
FILTER: mask(color=#EEEEEE)
                shadow(color=#66AAFF,direction=135)
                dropshadow(color=#6655FF,offx=-3,offy=-3,positive=2)
                chroma(color=#EEEEEE)
}
</style>
```

（2）在页面中添加表格，并应用以上定义的 CSS 样式，关键代码如下：

```
<table align="center"   height="60" class="tb1" border="0">
  <tr>
    <td height="60" align="center">明日科技论坛</td>
  </tr>
</table>
```

秘笈心法

IE 10 已经完成采用 HTML 5 标准了，它已经放弃了其特有的 Filters（滤镜）和 VML 了，取而代之的是 CSS 3 和 SVG。对于这两项内容希望大家能够自学一下，然后逐渐使用新的技术来取代之前经常应用的 Filters 和 VML。

实例 084　文字的阳文效果

光盘位置：光盘\MR\03\084

高级

实用指数：★★★★

实例说明

在设计网站时，为了体现网站的个性，可以设计网站内容文字的阳文效果。本实例将通过 CSS 样式的多种滤镜属性来实现，运行结果如图 3.32 所示。



图 3.32　文字的阳文效果

关键技术

本实例主要通过 CSS 样式的 mask 滤镜、dropshadow 滤镜和 chroma 滤镜实现。这些滤镜的用法在前面已经介绍过，在此不再赘述。

设计过程

（1）新建 index.htm 页面，编写文字阳文效果的 CSS 样式，关键代码如下：

```
<style type="text/css">
.tb1 {
 font-size: 50px;
 font-family: 隶书;
 FILTER: mask(color=white)
              dropshadow(color=royalblue,offx=-3,offy=-3,positive=1)
              chroma(color=white)
}
</style>
```

（2）在页面中添加表格，并应用 CSS 样式，关键代码如下：

```
<table align="center"   height="60" class="tb1" border="0">
  <tr>
    <td height="60" align="center">明日科技</td>
  </tr>
</table>
```

秘笈心法

对于本实例实现的这个文字的阳文效果，在 Firefox 或者 Chrome 浏览器中，可以通过 CSS 3 中的为文字指定多个阴影的 text-shadow 属性来实现，关键代码如下：

```
<style type="text/css">
div{
        text-shadow:1px 1px royalblue,
                              2px 2px royalblue,
                              3px 3px royalblue;
        color: white;
        font-size:50px;
        font-family:隶书;

</style>
```

实例 085　文字的雪雕效果

光盘位置：光盘\MR\03\085

高级　实用指数：★★★★

实例说明

在设计网站时，为了体现网站的个性，可以设计网站导航文字的雪雕效果。本实例将通过 CSS 样式的滤镜属性来设置文字的雪雕效果，运行结果如图 3.33 所示。



图 3.33　文字的雪雕效果

关键技术

本实例主要通过 CSS 样式的 glow 滤镜和 dropshadow 滤镜实现。先用 glow 滤镜使文字发光，然后使用 dropshadow 滤镜使文字具有阴影效果，再将这两个滤镜效果进行重叠即可。

设计过程

（1）新建 index.htm 页面，编写文字雪雕效果的 CSS 样式，关键代码如下：

```
<style type="text/css">
.tb1 {
 font-size: 40px;
 font-style: italic;
 font-family:楷体;
 filter: glow(color=#FFFFFF,strength=12)
                 dropshadow(color=#86BFFF,offx=3,offy=3,positive=1);
}
</style>
```

（2）在页面中添加表格，并应用以上定义的 CSS 样式，关键代码如下：

```
<table align="center" width="200" height="70" class="tb1" border="0">
   <tr>
      <td height="60" align="center">明日科技</td>
</tr>
</table>
```

秘笈心法

通过将 glow 滤镜和 dropshadow 滤镜效果进行重叠，便可实现文字的雪雕效果。为了达到更好的美化效果，可以将这样的字体样式设置为网站的字体样式。

实例 086　文字的火焰效果

光盘位置：光盘\MR\03\086

高级　实用指数：★★★★

实例说明

在设计网站时，为了体现网站的个性，可以设计网站内容文字的火焰效果。本实例将通过 CSS 样式的滤镜属性来设置文字的火焰效果，运行结果如图 3.34 所示。

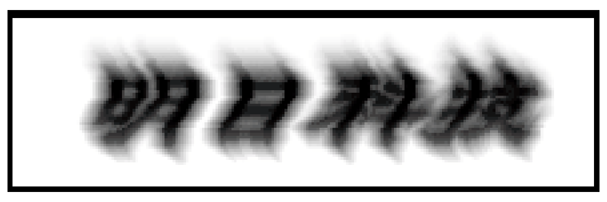


图 3.34　文字的火焰效果

关键技术

本实例主要通过 CSS 样式的 glow 滤镜、blur 滤镜和 wave 滤镜实现。

（1）blur 滤镜用于设置文字的模糊效果，语法格式如下：

```
{filter : blur(add = value,direction=value,strength=value)}
```

参数说明

❶ add：用来指定图片是否被改变成模糊效果，取值为 true 或 false。

❷ direction：用于设置模糊的方向，模糊效果是按顺时针方向进行的，0° 表示垂直向上，然后每 45° 为一个单位，默认值为 270° 。

❸ strength：表示有多少像素的宽度将受到模糊影响，默认为 5 个像素。

（2）wave 滤镜用于使文字出现波浪变形的效果，语法格式如下：

```
{filter : wave(add = value,freq=value,lightStrength=value,phase=value,strength=value)}
```

参数说明

❶ add：是否将元素按照波形样式打乱。

❷ freq：波纹的频率，即指定在一个元素上需要产生多少个完整的波纹。

❸ lightStrength：增强波纹的光影效果，取值范围在 0～100 之间。

❹ phase：设置正弦波的偏移量。

❺ strength：设置振幅大小。

设计过程

（1）新建 index.htm 页面，编写文字火焰效果的 CSS 样式，关键代码如下：

```
<style type="text/css">
.tb1 {
 font-size:40px;
 font-family:隶书;
 filter: glow(color=red,strength=6)
                  Blur(Add=true,Direction=0,Strength=10)
                  Wave(Add=add,Freq=3,LightStrength=10,Phase=5,Strength=5)
}
</style>
```

（2）在页面中添加表格，并应用文字火焰效果的 CSS 样式，关键代码如下：

```
<table align="center"   height="100" class="tb1" border="0">
   <tr>
      <td height="100" align="center">明日科技</td>
   </tr>
</table>
```

秘笈心法

在 CSS 3 中，实现 blur 滤镜的模糊效果可以通过 filter:blur 属性为实现，不过对于该属性，IE 10 还不支持，但是 Firefox、Chrome 和 Opera 浏览器已经支持了，只是在使用时，需要添加相应的前缀，Firefox 需要添加-moz-、Chrome 需要添加-webkit、Opera 需要添加前缀-o-。

实例 087	文字的扭曲动画 光盘位置：光盘\MR\03\087	高级 实用指数：★★★★

实例说明

在设计网站时，为了体现网站的个性，可以设计网站内容文字的扭曲动画效果。本实例将通过 CSS 样式的滤镜属性来设置文字的扭曲动画效果，运行结果如图 3.35 所示。

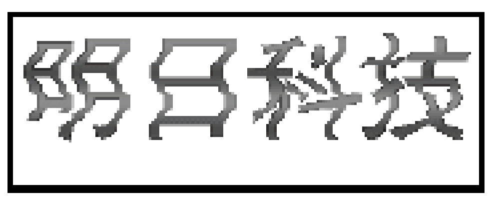


图 3.35　文字的扭曲动画

关键技术

本实例主要通过使用 CSS 样式的 wave 滤镜和 JavaScript 的 setTimeout()函数实现。

设计过程

（1）新建 index.htm 页面，编写文字扭曲效果的 CSS 样式，关键代码如下：

```
<style type="text/css">
.mydiv {
	width:300px;
	height:150px;
	font-family:黑体;
	font-size:50px;
	color:#00AAFF;
	filter:wave(freq=6, strength=4, phase=5, lightstrength=45, add=0, enabled=1)
}
</style>
```

（2）在页面中添加层，并应用以上定义的 CSS 样式，关键代码如下：

```
<div id="div1" class="mydiv">明日科技</div>
```

（3）在 JavaScript 方法中，使用 setTimeout()函数间隔指定的时间来修改 wave 滤镜的 phase 属性值（正弦波的偏移量），实现文字的扭曲动画，关键代码如下：

```
<script language="javascript">
var p=2;
function sport(){
	div1.filters.wave.phase=p;
	p=p+2;
setTimeout("sport()",20);
}
</script>
```

（4）在页面加载时调用 JavaScript 方法，实现文字的扭曲，关键代码如下：

```
<body onload="sport()">
```

秘笈心法

使用 CSS 样式中的 wave 滤镜还可以实现图片的水波纹效果，有兴趣的读者可以自己试试，也可以参照实例 092 来完成。

实例 088	输出文字 光盘位置：光盘\MR\03\088	高级 实用指数：★★★★

实例说明

本实例是将一组带阴影的文字在页面中一个一个地显示出来，并将该组文字以不同的语句在窗体上循环显示，运行结果如图 3.36 所示。

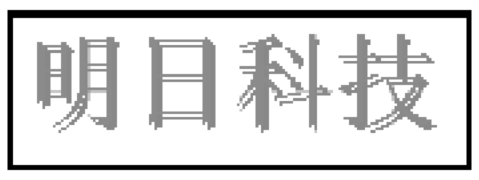


图 3.36　输出文字

关键技术

本实例主要是用<span >标记中的 style 样式的 clip 属性的 rect()方法来实现字符串的输出效果，下面对 rect()方法进行详细说明。

rect()方法的语法格式如下：

```
clip : auto | rect( number number number number )
```

参数说明

❶ auto：对象无剪切。

❷ rect(number number number number)：依据上-右-下-左的顺序提供指定对象左上角为（0,0）坐标的 4 个偏移数值，其中任一数值都可用 auto 替换，即此边不剪切。

功能：检索或设置对象的可视区域。区域外的部分是透明的，必须将 position 的值设为 absolute，此属性方可使用。

设计过程

（1）在页面中添加两个层，shadow 层用于显示实体的文字，showstr 层用于显示文字的阴影，代码如下：

```
<div id="shadow" style="position:absolute;visibility:visible;top:12px; left:12px"></div>
<div id="showstr" style="position:absolute;visibility:visible;top:10px; left:10px"></div>
```

（2）编辑用于实现输出文字的 JavaScript 代码。

用一维数组来保存要进行输出的字符串，代码如下：

```
<script language="JavaScript">
        var space=0
        var word=0
        var Tstr = new Array()
        Tstr[0]="明日科技"
        Tstr[1]="欢迎购买明日科技图书"
        Tstr[2]="明日科技网址：www.mingrisoft.com"
</script>
```

自定义函数 exportstr()，以输出的方式循环显示 Tstr 数组中的字符串，代码如下：

```
<script language="JavaScript">
function exportstr() {
if (space <= document.body.clientWidth){
    showstr.innerHTML="<span style='position:absolute;font-family:宋体;
        font-size:30pt;color:3399FF; clip:rect(2px "+space+"px 200px 0px)'>"+Tstr[word]+"</span>"
    shadow.innerHTML="<span style='position:absolute;font-family:宋体;
        font-size:30pt;color:888888; clip:rect(2px "+space+"px 200px 0px)'>"+Tstr[word]+"</span>"
    space=space+30
    var timer=setTimeout("exportstr()",30)
}
else {
    document.close()
    clearTimeout(timer)
    word++
    space=0
    if (word >= Tstr.length) {word=0}
    var intermezzo=setTimeout("exportstr()",100)
}
}
exportstr();
</script>
```

秘笈心法

setTimeout()函数的作用是在指定的时间过后，调用 JavaScript 代码或自定义的函数，并且只被调用一次。如果希望一直调用某个 JavaScript 方法，需要将 setTimeout()函数写在被调用的 JavaScript 方法体内。

3.6　图片滤镜特效

在设计网页时，经常需要在网页中添加图片，而且会对图片进行特效处理，这样会使整个网页看起来更加美观。本节将通过几个实例讲解如何利用 CSS 样式中的滤镜属性来处理图片的特效。

实例 089	图片的半透明效果 光盘位置：光盘\MR\03\089	高级 实用指数：★★★★

实例说明

在设计网站时，为了美化网站，可以把网站中添加的图片制作成半透明效果。本实例将通过 CSS 样式来设置图片的半透明效果，运行结果如图 3.37 所示。



图 3.37　图片的半透明效果

关键技术

本实例主要通过 CSS 样式的 alpha 滤镜属性实现，该滤镜用于设置图片的透明度，其语法格式如下：

```
{filter: glow(opacity= value,finishOpacity=value,style=value,startX=value,startY=value,finishX=value,finishY=value)}
```

参数说明

❶ opacity：表示图片的透明程度。取值范围在 0～100 之间，0 表示完全透明，100 表示完全不透明。

❷ finishOpacity：设置渐变的透明效果时，用该参数指定结束时的透明度，取值范围在 0～100 之间。

❸ style：指定透明区域的形状特征，0（统一形状）、1（线形）、2（放射状）、3（长方形）。

❹ startX：渐变透明效果开始的 x 坐标。

❺ startY：渐变透明效果开始的 y 坐标。

❻ finishX：渐变透明效果结束的 x 坐标。

❼ finishY：渐变透明效果结束的 y 坐标。

设计过程

（1）新建 index.htm 页面，编写图片透明效果的 CSS 样式，关键代码如下：

```
<style type="text/css">
.transparency {
        filter:Alpha(Opacity=10,finishOpacity=80,style=3,startX=20,startY=20,finishX=100,finishY=100)
}
</style>
```

（2）在页面中添加表格，在表格中添加两个用于显示图片的<img>标签，其中一个<img>标签使用以上定义的 CSS 样式，关键代码如下：

```
<table   border="0" cellpadding="0" cellspacing="0" >
  <tr>
        <td><img src="img1.jpg" width="378" height="275"></td>
        <td> </td>
     <td><img src="img1.jpg" width="378" height="275" border="0" class="transparency"></td>
  </tr>
</table>
```

秘笈心法

为了提高开发效率，可以将 alpha 滤镜的样式代码写到.css 样式文件中，在图片需要加入特效时，直接通过<link>标签引用 CSS 样式文件即可，避免每次都编写图片滤镜的特效代码。

实例 090	图片的模糊效果 光盘位置：光盘\MR\03\090	高级 实用指数：★★★★

实例说明

在设计网站时，为了美化网站，有时需要把网站中添加的图片制作成模糊效果。本实例将通过 CSS 样式来设置图片的模糊效果，运行效果如图 3.38 所示。



图 3.38　图片的模糊效果

关键技术

本实例主要应用 CSS 样式的 blur 滤镜属性实现，blur 滤镜的用法请参考实例 086。

设计过程

（1）新建 index.htm 页面，编写图片模糊效果的 CSS 样式，关键代码如下：

```
<style>
.blurStyle {
	filter:blur(add=true,direction=270,strength=20)
}
</style>
```

（2）在页面中添加<img>标签，并使用以上定义的 CSS 样式，关键代码如下：

```
<table>
 <tr>
	<td><img src="img1.jpg" width="400" height="300"></td>
	<td> </td>
	<td><img class="blurStyle" src="img1.jpg" width="400" height="300"></td>
 </tr>
</table>
```

秘笈心法

为了提高开发效率，可以将 blur 滤镜的样式代码写到.css 样式文件中，在图片需要加入特效时，直接通过<link>标签引用 CSS 样式文件即可，避免每次都编写图片滤镜的特效代码。

实例 091	图片的渐隐渐现效果 光盘位置：光盘\MR\03\091	高级 实用指数：★★★★

实例说明

在设计网站时，为了美化网站，有时需要把网站中添加的图片制作成渐隐渐现效果。本实例将通过 JavaScript 方法以及 CSS 样式使图片具有渐隐渐现的动态效果，运行效果如图 3.39 所示。



图 3.39　图片的渐隐渐现效果

关键技术

实现本实例，首先需要通过 CSS 样式的 alpha 滤镜对图片设置一个初始状态的透明度，然后在 JavaScript 方法中通过 setTimeout()函数间隔指定时间来修改图片对象的滤镜属性中的透明度属性值，这样便实现了图片的渐隐渐现效果。

设计过程

（1）新建 index.htm 页面，编写图片透明度效果的 CSS 样式，关键代码如下：

```
<style type="text/css">
.transparency {
	filter:Alpha(Opacity=5,finishOpacity=80,style=3,startX=20,startY=20,finishX=100,finishY=100)
}
</style>
```

（2）在页面中添加用于显示图片的<img>标签，设置<img>标签的 id 属性值，并应用以上定义的 CSS 样式，关键代码如下：

```
<img src="img1.jpg" id="myimg" width="378"   height="275" border="0" class="transparency">
```

（3）编写 JavaScript 方法，实现图片的渐隐渐现效果，关键代码如下：

```
<script type="text/javascript">
        var i=5;
        function init(){
                document.all.myimg.filters.alpha.opacity=i;            //给图片对象滤镜中的透明度属性赋值
                i+=5;
                if(i==100){                                             //当表示透明度的 i=100 时，重新指定 i=5
                        i=5;
                }
                setTimeout("init()",100);                               //每隔 100 毫秒，调用一次 init()方法改变图片透明度
        }
</script>
```

（4）在页面加载时，调用 JavaScript 方法初始化图片的透明度，关键代码如下：

```
<body onload="init()">
```

秘笈心法

本实例应用到了 setTimeout()函数，该函数的用法与 setInterval()函数比较类似，它们都可以在指定的时间内调用 JavaScript 代码或 JavaScript 自定义方法。不过这两个函数是有区别的，使用 setTimeout()函数时，只会执行一次所调用的 JavaScript 代码，如果想循环执行 JavaScript 代码，必须将 setTimeout()函数写在被调用的 JavaScript 自定义函数体内。但是 setInterval()函数则不同，通过它会一直循环执行 JavaScript 代码，并不需要将 setInterval()函数写在 JavaScript 方法体内。

实例 092　图片的水波纹效果

光盘位置：光盘\MR\03\092

高级

实用指数：★★★★

实例说明

在设计网站时，为了美化网站，有时需要把网站中添加的图片制作成水波纹效果。本实例将通过 CSS 样式来实现图片的水波纹效果，运行效果如图 3.40 所示。



图 3.40　图片的水波纹效果

关键技术

本实例主要通过 CSS 样式的 wave 滤镜实现，wave 滤镜的用法在实例 086 中已经介绍过，在此不再赘述。

设计过程

（1）新建 index.htm 页面，编写图片水波纹效果的 CSS 样式，关键代码如下：

```
<style type="text/css">
.ripple {
        filter:wave(freq=17, strength=3, phase=7, lightstrength=35, add=0, enabled=1);
}
</style>
```

（2）在页面中添加用于显示图片的<img>标签，并应用以上定义的 CSS 样式，关键代码如下：

```
<img class="ripple" src="fj.JPG" width="400" height="300" >
```

秘笈心法

为了提高开发效率，可以将 wave 滤镜的样式代码写到.css 样式文件中，在图片需要加入特效时，直接通过<link>标签引用 CSS 样式文件即可，避免每次都编写图片滤镜的特效代码。

实例 093 图片的灰度效果

光盘位置：光盘\MR\03\093

高级

实用指数：★★★★

实例说明

在设计网站时，为了美化网站，有时需要把网站中添加的图片制作成灰度效果。本实例将通过 CSS 样式来设置图片的灰度效果，运行效果如图 3.41 所示。



图 3.41　图片的灰度效果

关键技术

本实例主要通过 CSS 样式的 gray 滤镜实现，其语法格式如下：

```
{filter: gray}
```

设计过程

（1）新建 index.htm 页面，编写图片灰度效果的 CSS 样式，关键代码如下：

```
<style type="text/css">
.old {
        filter:gray;
}
</style>
```

（2）在页面中添加一个<img>标签，并且应用以上定义的 CSS 样式，关键代码如下：

```
<img class="old" src="fj.JPG" width="400" height="300" >
```

秘笈心法

为了提高开发效率，可以将 gray 滤镜的样式代码写到.css 样式文件中，在图片需要加入特效时，直接通过<link>标签引用 CSS 样式文件即可，避免每次都编写图片滤镜的特效代码。

实例 094 图片的动态说明文字

光盘位置：光盘\MR\03\094

高级

实用指数：★★★★

实例说明

本实例将通过 CSS 样式以及 JavaScript 方法来实现图片的动态文字的效果。当鼠标经过图片时，会显示图片的说明文字，并且将图片设置为半透明效果，运行效果如图 3.42 所示。



图 3.42　图片的动态说明文字效果

关键技术

本实例的实现应用了多种 CSS 样式，首先定义一个不透明的和一个透明度为 50%的 alpha 滤镜样式，然后定义图片说明文字显示和隐藏以及说明文字位置的样式，最后通过鼠标事件调用 JavaScript 方法来动态改变 CSS 样式。

设计过程

（1）新建 index.htm 页面，编写图片透明度以及图片说明文字的 CSS 样式，关键代码如下：

```
<style type="text/css">
.style1 {
	position: absolute;
	display: none;
}
.style2 {
	position:relative;
	position: absolute;
	display:block;
}
.style3{
	filter:alpha(opacity=50,finishOpacity=20,style=3,startX=10,startY=5,finishX=90,finishY=20);
}
.style4{
	filter:alpha(opacity=100,finishOpacity=100,style=3,startX=10,startY=5,finishX=90,finishY=20);
}
</style>
```

（2）编写鼠标经过事件所调用的 JavaScript 方法，关键代码如下：

```
<script type="text/javascript">
function over(){
	document.all.text1.className="style2";
	document.all.myimg.className="style3";
}function out(){
	document.all.text1.className="style1";
	document.all.myimg.className="style4";
}
</script>
```

（3）在页面中添加<img>标签和用于显示说明文字的<span>标签，并设置<img>标签的 onMouseover（鼠标经过事件）属性和 onMouseout（鼠标移出事件）属性，关键代码如下：

```
<span id="text1" class="style1">
	<font color="green">这是一个风景图片</font>
</span>
<img class="style4" src="fj.jpg" id="myimg" width="200"   height="150" border="0" onmouseover="over()" onmouseout="out()">
```

秘笈心法

为了提高开发效率，可以将图片滤镜的样式代码写到.css 样式文件中，在图片需要加入特效时，直接通过<link>标签引用 CSS 样式文件即可，避免每次都编写图片滤镜的特效代码。

第 4 章

JSP 基础与内置对象

- JSP 的基本应用
- JSP 内置对象
- JSP 的自定义标签

4.1 JSP 的基本应用

JSP 是 Java Server Page 的缩写，它是 Servlet 的扩展，是一种基于文本的程序，在 Java Web 应用开发时是必不可少的。JSP 的特点是 HTML 代码与 Java 程序共同存在，在接收到用户请求时，服务器会处理 Java 代码片段，然后生成处理结果的 HTML 页再返回给客户端，客户端的浏览器将呈现最终页面效果。

实例 095 自定义错误页面

光盘位置：光盘\MR\04\095

初级

实用指数：★

实例说明

像 Java 程序一样，JSP 在运行时也有可能抛出异常，因此为了使界面更加友好，需要自定义错误页面，应用 JSP 指令，当在 JSP 中抛出异常时，自动导向错误页面。本实例将介绍如何应用 JSP 自定义错误页面，实例运行结果如图 4.1 所示。

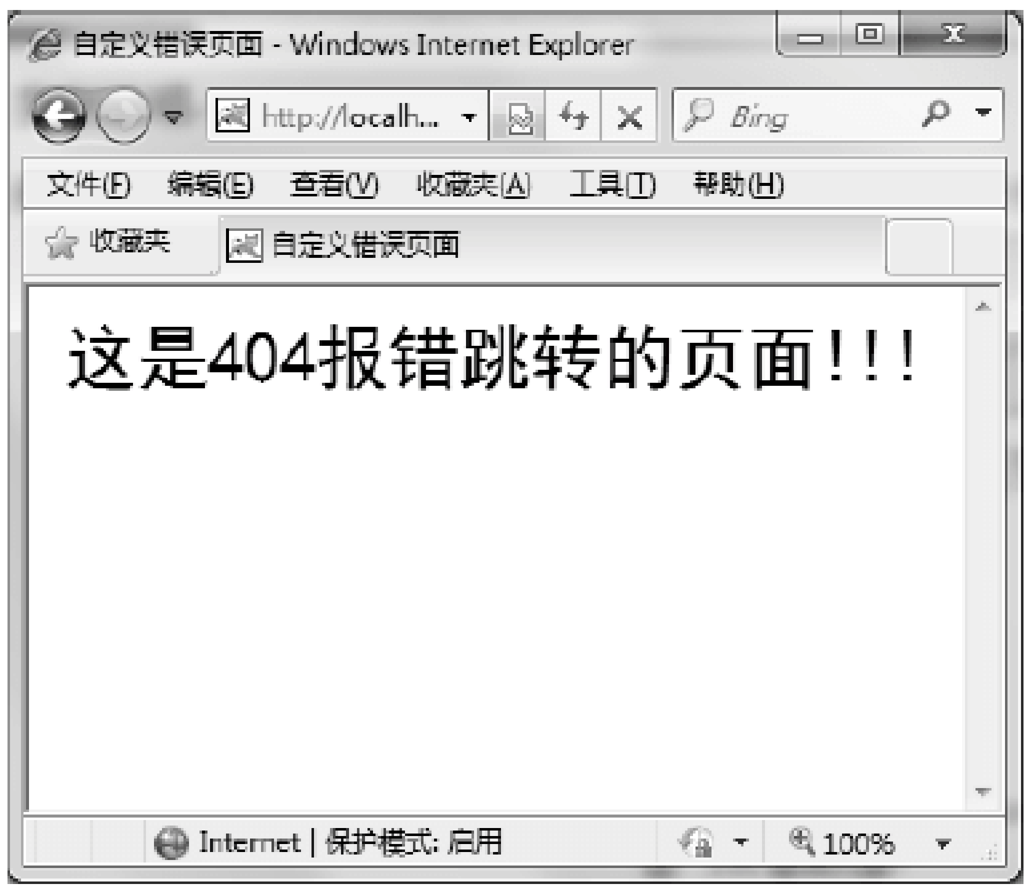


图 4.1 自定义错误页面

关键技术

本实例主要使用 JSP 中的 page 指令的 errorPage 属性自定义错误页面，代码如下：

```
<%@ page errorPage="error.jsp" %>
```

其中 error.jsp 是一个专门负责处理异常的网页。

设计过程

（1）创建 error.jsp 页面，在 page 指令中配置 errorPage 属性，errorPage 属性值就是当发生异常时跳转的页面，关键代码如下：

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"% errorPage="error.jsp"%>
```

（2）在 error.jsp 中，通过如下语句将该网页声明为异常处理网页。

```
<% page isErrorPage="true"%>
```

秘笈心法

也可以应用 web.xml 文件中的 error-page 参数定义错误页面，应用该参数配置的错误页面将对所有页面有效，也就是说，无论是在哪个页面中出现了异常，都会被导向 error-page 参数所设置的错误页面，代码如下：

```
<error-page>
        <error-code>404</error-code>
        <location>/error.jsp</location>
 </error-page>
```

实例 096 导入版权信息

光盘位置：光盘\MR\04\096

初级

实用指数：★★

实例说明

在浏览网站时，通常在网页底端会标明网站的版权所有信息，而且网站中每个网页都包含相同内容的版权信息。在应用 JSP 技术开发网站时，也需要为网站添加版权信息。本实例将介绍如何应用 JSP 指令包含指定的版权信息，运行结果如图 4.2 所示。



图 4.2　在网站的底端包含的版权信息

关键技术

在 JSP 中可以应用<jsp:include>标签包含其他文件的内容，被包含的文件可以是 JSP 文件或 HTML 文件，<jsp:include>标签的语法如下：

```
<jsp:include page="被包含组件的绝对 URL 或相对 URL">
```

设计过程

（1）创建一个包含版权信息的页面 foot.jsp，关键代码如下：

```
<body>
  <table border="0" align="center">
      <tr>
            <td align="center">编程词典销售服务热线：400-675-1000  网址：www.mingrisoft.com</td>
      </tr>
      <tr>
            <td align="center">Copyright@www.mingrisoft.com All Rights Reserved!</td>
      </tr>
  </table>
</body>
```

（2）创建网站的主页面 index.jsp，在该页中应用<jsp:include>标签将版权信息页 foot.jsp 包含进来，关键代码如下：

```
<body>
<table>
……此处省略了其他标签的代码
<tr>
      <td colspan="3">
            <jsp:include page="foot.jsp"></jsp:include>
      </td>
```

```
</tr>
</table>
</body>
```

秘笈心法

在开发网站时，如果多数网页都包含相同的内容，可以把这部分内容单独放到一个文件中，其他的 JSP 文件通过 include 标签即可将这个文件包含进来。这样做可以提高代码的可重用性，避免重复代码，从而提高开发网站的效率。

实例 097　应用 Java 程序片段动态生成表格

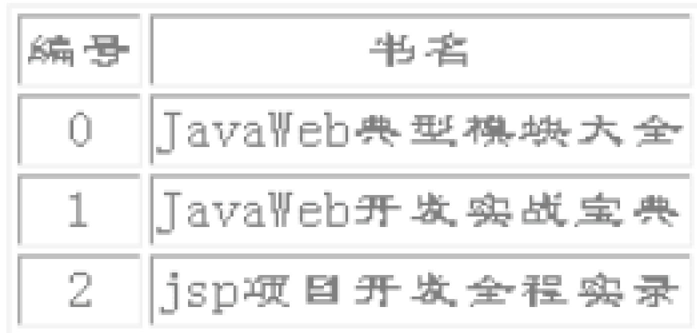
光盘位置：光盘\MR\04\097

初级

实用指数：★★

实例说明

在实际的项目开发过程中，经常需要通过 JSP 页面动态显示数据库中的数据，如在根据条件进行查询时，需要将查询结果显示在 JSP 页面中，此时需要应用表格来显示查询结果，由于查询结果是动态的，数据行数并不固定，因此在显示查询结果数据时，要根据数据的行数来确定表格的行数。本实例将介绍如何应用 Java 程序片段（Scriptlet）动态生成表格，运行结果如图 4.3 所示，表格中的后 3 行是动态生成的。



编号	书名
0	JavaWeb典型模块大全
1	JavaWeb开发实战宝典
2	jsp项目开发全程实录

图 4.3　应用 Java 程序片段动态生成表格

关键技术

在 JSP 文件中，可以在“<%”和“%>”标记中直接嵌入任何有效的 Java 程序代码，这种嵌入的程序片段称为 Scriptlet。应用这个标记，就可以随时在 JSP 中嵌入 Java 程序代码。因此实现本实例非常简单，只要在 JSP 页面的相应位置应用“<%”和“%>”标记即可动态生成表格。

设计过程

（1）创建 index.jsp 页面，首先在“<%”和“%>”中定义一个字符串数组，用于存储图书名称字符串，关键代码如下：

```
<%
    String[] bookName = {"JavaWeb 典型模块大全","JavaWeb 开发实战宝典" ,"jsp 项目开发全程实录"};
 %>
```

（2）在 index.jsp 页的表格处，循环步骤（1）中创建的字符串数组，动态生成表格并将数据添加到表格中，关键代码如下：

```
<table border="1" align="center">
    <tr>
        <td align="center">编号</td>
        <td align="center">书名</td>
    </tr>
    <%
        for(int i=0;i<bookName.length;i++){
     %>
    <tr>
        <td align="center"><%=i %></td>
        <td align="center"><%=bookName[i]%></td>
    </tr>
    <%} %>
</table>
```

秘笈心法

从本实例步骤（2）的代码中可以看出，插入在 JSP 页面中的一个完整的 Java 程序片段，可以应用<%和%>

分成多个部分，在第一处的<%和%>标记中的 for 循环没有返回大括号}，而是在最后</table>之上添加的，这样，在 for 循环中，<tr>和<td>标签将作为循环体的内容被输出。这段代码在编译后，JSP 会生成相应的 Servlet 文件。

实例 098　应用 Java 程序片段动态生成下拉列表

光盘位置：光盘\MR\04\098

初级

实用指数：★★

实例说明

在从数据库中读取数据时，有时需要将某些数据动态添加到下拉列表中。如在根据条件查询数据时，需要根据下拉列表的选项值查询数据库，那么这个下拉列表的值就必须保证是从数据库中读取出来的，也就是说只有下拉列表的选项必须是数据库中存在的数据才可以进行查询。所以，有时可能需要通过 Java 程序片段动态地将数据添加到下拉列表中。运行本实例，如图 4.4 所示，下拉列表中的数据是动态添加的。

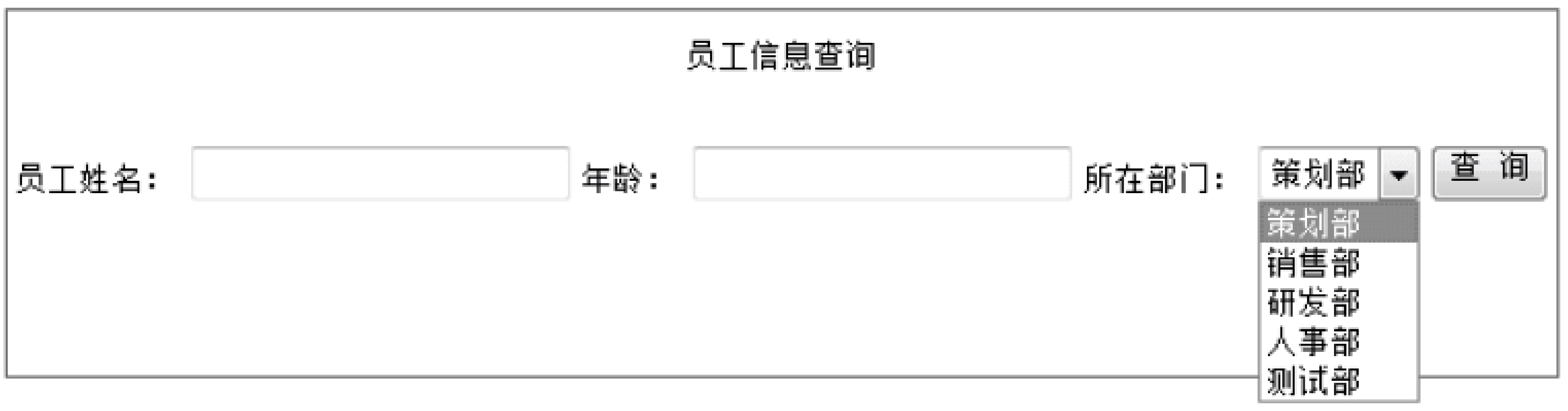


图 4.4　动态生成下拉列表

关键技术

在实现本实例时，并没有进行读取数据库，只是在“<%”和“%>”的标记中创建了一个字符串数组，然后将这个数组数据添加到下拉列表中。

提示：其实这样和读取数据库中的数据再添加到下拉列表中并没有多大区别，本实例只是为了演示如何将数据添加到下拉列表中，并没有读取数据库。读者在实现时，可以实现从数据库中读取数据并将其添加到下拉列表中。

设计过程

（1）新建 index.jsp 页，首先在该页中创建一个字符串数组，用于保存员工的部门信息，关键代码如下：

```
<%
        String[] dept = {"策划部","销售部","研发部","人事部","测试部"};
%>
```

（2）在下拉列表处，循环这个数组，将数组中的元素添加到下拉列表中，关键代码如下：

```
<table height="150">
        <tr>
                <td align="center" colspan="4">员工信息查询</td>
        </tr>
        <tr>
                <td>员工姓名：<input type="text" name="name" /></td>
                <td>年龄：<input type="text" name="age" /></td>
                <td>所在部门：
                        <select>
                                <%
                                        for(int i=0;i<dept.length;i++){%>
                                                <option value="<%=dept[i] %>"><%=dept[i] %></option>
                                         <%}
                                %>
                        </select>
                </td>
                <td><input type="button" value="查  询" /></td>
```

```
        </tr>
        <tr>
                <td align="center" colspan="4" height="50"></td>
        </tr>
</table>
```

秘笈心法

在下拉列表的选项中，应用的是 Java 表达式标记输出的数组元素，Java 表达式的标记为“<%=”和“%>”，如果在 JSP 文件的模板文本中使用该标记，那么它能把表达式的值输出到网页上。

实例 099　同一页面中的多表单提交

光盘位置：光盘\MR\04\099

初级

实用指数：★★

实例说明

在编写程序时，会遇到同一个页中多个表单提交的情况。处理同一个页中多个表单的提交主要是为每个表单提供相应的标识，当表单提交后，根据传递的标识来判断提交的表单，并执行相应的处理。运行本实例，在“表单 1”文本框中输入“豆豆上线”，在“表单 2”文本框中输入“无语上线”，在“表单 3”文本框中输入“林夕上线”后，如图 4.5 所示；单击“表单 2”后面的“提交”按钮，即可显示如图 4.6 所示的页面。



图 4.5　表单提交前的页面

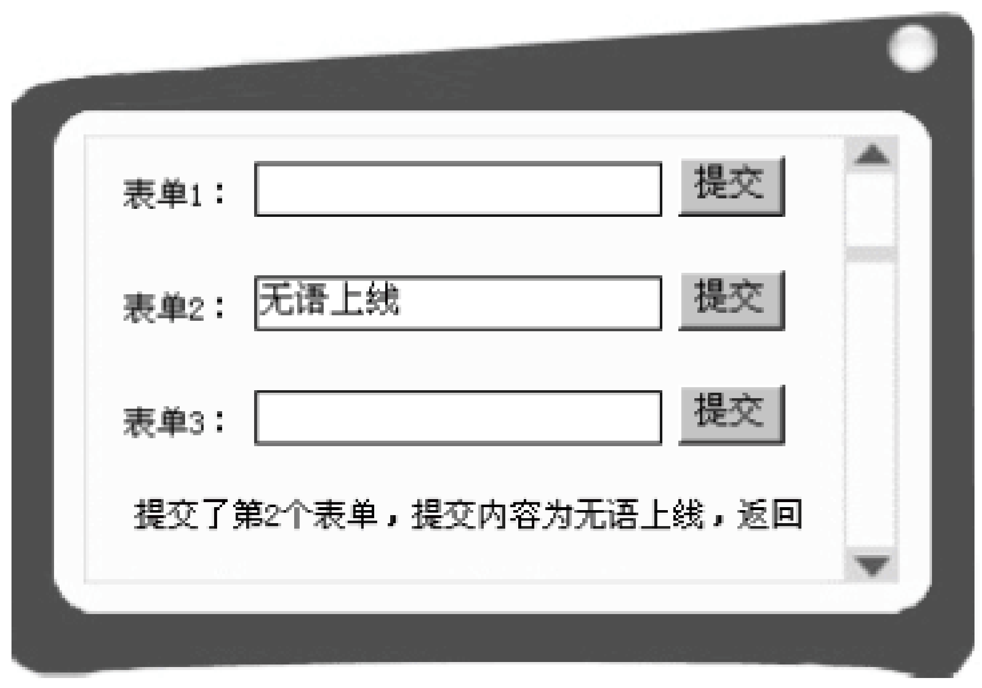


图 4.6　表单 2 提交后的页面

关键技术

本实例主要应用参数的传递，通过传递参数值的不同获取不同的信息。首先设置 3 个显示表单数据的对象（tex1,tex2,tex3），并且设置其初始值为空字符串，其次通过 request.getParameter()方法请求对应的表单参数，如果请求的参数不为空字符串，则将参数赋给相应的对象，最后通过表单元素显示提交后相应的信息。

设计过程

（1）通过 JavaScript 脚本限定文本字段不允许其为空值，关键代码如下：

```
<script language="javascript">
function Mycheck(){
        if(form1.text1.value==""){
                alert("请输入表单的内容！！！ ");form1.text1.focus();return;
        }
        form1.submit();
}
</script>
<script language="javascript">
function Mycheck1(){
        if(form1.text1.value==""){
                alert("请输入表单的内容!!");form1.text1.focus();return;}
        form1.submit();
}
</script>
<script language="javascript">
```

```
function Mycheck2(){
        if(form3.text3.value==""){
                alert("请输入表单的内容!!!");form3.text3.focus();return;}
        form3.submit();
}
</script>
```

（2）设置 Form 表单和文本字段的相关属性，关键代码如下：

```
<form name="form1" method="post" action="?formid=1">
表单 1:
       <input name="text1" type="text" class="text" value="<%=text1%>">
       <input type="button" name="submit1" value="提交" onclick="Mycheck();">
</form>
<form name="form2" method="post" action="?formid=2">
表单 2:
       <input name="text2" type="text" class="text" value="<%=text2%>">
       <input type="button" name="submit2" value="提交" onclick="Mycheck1();">
</form>
<form name="form3" method="post" action="?formid=3">
表单 3:
       <input name="text3" type="text" class="text" value="<%=text3%>">
       <input type="button" name="submit3" value="提交" onclick="Mycheck2();">
</form>
```

（3）表单提交后，根据提交的表单获取相关信息，关键代码如下：

```
<%
String text1="";
String text2="";
String text3="";
String message="";
if(request.getParameter("text1")!=null){
        text1=request.getParameter("text1");
        message="提交了第 1 个表单，提交内容为"+text1+"";
}
if(request.getParameter("text2")!=null){
        text2=request.getParameter("text2");
        message="提交了第 2 个表单，提交内容为"+text2+"";
}
if(request.getParameter("text3")!=null){
        text3=request.getParameter("text3");
        message="提交了第 3 个表单，提交内容为"+text3+"";
}
%>
```

秘笈心法

应用参数传递时，如果是提交到本页，也可以省略文件名直接输入问号“?”加参数和值进行参数传递，如 action="?formid=1"。

实例 100 在 JSP 脚本中插入 JavaScript 代码

光盘位置：光盘\MR\04\100

初级

实用指数：★★

实例说明

所谓的 JSP 脚本，也就是前几个实例中用到的 Java 程序片段（Scriptlet）。在实际应用中，有时需要在 JSP 脚本中插入 JavaScript 代码，如在一些网站中添加注册信息时，注册成功后可能需要弹出一个“注册成功”的提示信息窗口，这就需要在 JSP 脚本中插入 JavaScript 的 alert()方法来实现。运行本实例，如图 4.7 所示，输入用户注册信息，单击“注册”按钮后，将弹出注册成功的提示框。

关键技术

在 JSP 脚本中插入 JavaScript 代码，可以应用 JSP 中的 out 隐含对象，通过 out 对象可以向客户端浏览器输

出信息，并且管理应用服务器上的输出缓冲区。在使用 out 对象输出数据时，可以对数据缓冲区进行操作，及时清除缓冲区中的残余数据，为其他的输出让出缓冲空间。待数据输出完毕后，要及时关闭输出流。



图 4.7　在 JSP 脚本中插入 JavaScript 代码

out 对象一个最基本的应用就是向客户端浏览器输出信息。out 对象可以输出各种数据类型的数据，在输出非字符串类型的数据时，会自动转换为字符串进行输出。out 对象提供了 println()方法向页面中输出信息的方法，应用该方法可以直接输出 HTML 标签，因此，可以应用它来输出<script>和</script>标签之间的 JavaScript 代码，其代码如下：

```
<%
	out.println("<script> alert('输出弹出提示窗口！');<script>")
%>
```

注意：应用 out 对象的 println()方法输出 HTML 标签时，是以字符串格式输出的，如果输出标签体的属性信息，需要应用转义字符 “\”，如输出一个文本框元素：out.println("<input type=\"text\" name=\"user\"/>")。

设计过程

（1）创建用户注册的表单页 index.jsp，将表单提交到 save.jsp 页，关键代码如下：

```
<form action="save.jsp" method="post">
	……
</form>
```

（2）创建 save.jsp 页，该页只是为了演示如何在 JSP 脚本中插入 JavaScript 代码，并没有真正对表单数据进行处理，如何获取表单数据，会在后面的实例中具体介绍。save.jsp 的关键代码如下：

```
<%
	out.println("<script> alert('注册成功！');window.location.href='index.jsp';</script>");
%>
```

秘笈心法

out 对象的类一个比较重要的功能就是对缓冲区进行管理。通过调用 out 对象的 clear()方法可以清除缓冲区的内容。这类似于重置响应流，以便重新开始操作。如果响应已经提交，则会产生 IOException 异常的负作用。out 对象还提供了另一种清除缓冲区内容的方法，即 clearBuffer()方法，通过该方法可以清除缓冲区的“当前”内容，而且即使内容已经提交给客户端，也能够访问该方法。

实例 101	将页面转发到用户登录页面 光盘位置：光盘\MR\04\101	初级 实用指数：★★

实例说明

通过<jsp:forward>动作标识可以将请求转发到其他的 Web 资源，如另一个 JSP 页面、HTML 页面、Servlet 等。执行请求转发后，当前页面将不再被执行，而是去执行该标识指定的目标页面。运行本实例，将显示如图 4.8 所示的用户登录页面。

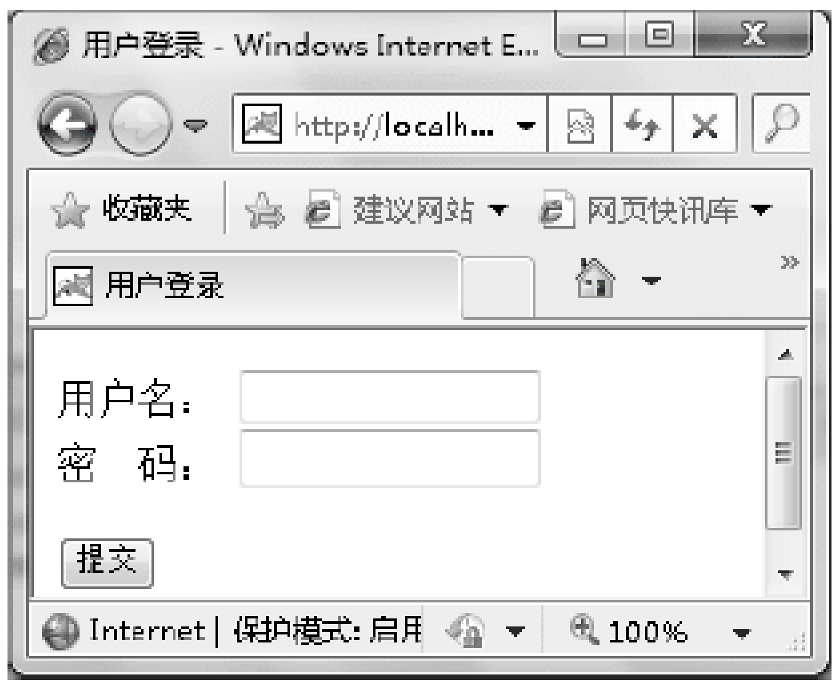


图 4.8　将页面转发到用户登录页面

关键技术

<jsp:forward>动作标识的语法格式如下：

```
<jsp:forward page="url"/>
```

或

```
<jsp:forward page="url">
 子动作标识<jsp:param>
</jsp:forward>
```

page 属性：用于指定请求转发的目标页面。该属性值可以是一个指定文件路径的字符串，也可以是表示文件路径的 JSP 表达式。但是请求被转向的目标文件必须是内部的资源，即当前应用中的资源。

JSP 的动作标识<jsp:param>可以作为其他标识的子标识，用于为其他标识传递参数，其语法格式如下：

```
<jsp:param name="参数名" value="参数值" />
```

参数说明

❶ name：用于指定参数名称。

❷ value：用于设置对应的参数值。

例如，通过<jsp:param>标识为<jsp:forward>标识指定参数，可以使用下面的代码：

```
<jsp:forward page="modify.jsp">
 <jsp:param name="userId" value="7"/>
</jsp:forward>
```

在上面的代码中，实现了在请求转发到 modify.jsp 页面的同时，传递了参数 userId，其参数值为 7。

设计过程

（1）创建一个名称为 index.jsp 的文件，该文件为中转页，用于通过<jsp:forward>动作标识将页面转发到用户登录页面（login.jsp），其具体代码如下：

```
<%@ page language="java" contentType="text/html; charset=GB18030" pageEncoding="GB18030"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=GB18030">
<title>中转页</title>
</head>
<body>
<jsp:forward page="login.jsp"/>
</body>
</html>
```

（2）编写 login.jsp 文件，在该文件中添加用于收集用户登录信息的表单及表单元素，具体代码如下：

```
<%@ page language="java" contentType="text/html; charset=GB18030" pageEncoding="GB18030"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=GB18030">
<title>用户登录</title>
</head>
<body>
<form name="form1" method="post" action="">
用户名：  <input    name="name" type="text" id="name" style="width: 120px"><br>
密  码：  <input name="pwd" type="password" id="pwd" style="width: 120px"> <br>
<br>
```

```
<input type="submit" name="Submit" value="提交">
</form>
</body>
</html>
```

秘笈心法

通过<jsp:param>动作标识指定的参数，将以“参数名=值”的形式加入到请求中。它的功能与在文件名后面直接加“?参数名=参数值”是相同的。

4.2　JSP 内置对象

JSP 为了简化开发，提供了一些内置对象，用来实现很多 JSP 应用。在使用 JSP 内置对象时，不需要先定义这些对象，直接使用即可。JSP 提供了 request、response、session、application、out、page、config、exception 和 pageContent 9 个内置对象。其中 request 对象、response 对象、out 对象、page 对象、config 对象、exception 对象和 pageContent 对象的有效范围是当前页面，而 session 对象的有效范围是当前会话，即同一个客户端的所有页面；application 对象的有效范围是当前应用，即在同一个应用程序中，只要服务器不关闭，这个对象就有效。

JSP 内置对象是应用 JSP 进行 Web 开发时经常被使用的技术。通过它们可以对 Web 开发中的请求、响应、会话、应用、输出、配置信息和异常信息等内容进行控制。对于不同的内容将由不同的内容进行控制。下面将介绍各内置对象的使用场合。

实例 102　获取表单提交的信息

光盘位置：光盘\MR\04\102

初级　实用指数：★★

实例说明

在 Web 应用程序中，经常需要完成用户与网站的交互。例如，当用户填写表单后，需要把数据提交给服务器处理，这时服务器就需要获取这些信息。本实例将介绍如何获取表单提交的信息。运行本实例，如图 4.9 所示，输入用户注册信息，单击“注册”按钮后，在 JSP 处理页中将获取到这些表单数据，并将其显示在页面中，如图 4.10 所示。



图 4.9　用户注册表单页



图 4.10　获取用户注册信息

关键技术

JSP 的内置对象 request 封装了由客户端生成的 HTTP 请求的所有细节，主要包括 HTTP 头信息、系统信息、请求方式和请求参数等。通过 request 对象提供的相应方法可以处理客户端浏览器提交的 HTTP 请求中的各项参数。

在本实例中，通过 request 对象的 getParameter()方法可以获取用户提交的表单信息。例如，存在一个 name

属性为 username 的文本框，在表单提交后，要获取其 value 值，可以通过下面的代码实现：

```
String userName = request.getParameter("username");
```

参数 username 与 HTML 表单的 name 属性对应，如果参数值不存在，则返回一个 null 值，该方法的返回值为 String 类型。

设计过程

（1）创建用户注册的表单页 index.jsp。

（2）创建 save.jsp 页面，在该页面中获取用户提交的表单信息并显示在页面中，关键代码如下：

```
<%
 request.setCharacterEncoding("UTF-8");
 String name=request.getParameter("name");
 String sex = request.getParameter("sex");
 String age = request.getParameter("age");
 String email=request.getParameter("email");
  %>
<table height="150" width="300">
    <tr><td align="center" colspan="4" height="20"></td></tr>
    <tr><td align="center" colspan="4">获取表单信息</td></tr>
    <tr>
        <td>用户名：</td>
        <td>
                <%if(name==null||"".equals(name)){
                        out.println("");
                }else{
                        out.println(name);
                } %>
        </td>
    </tr>
    <tr>
    <td>性别：  </td>
    <td>
        <%if(sex==null||"".equals(sex)){
                out.println("");
        }else{
                out.println(sex);
                } %>
        </td>
    </tr>
    <tr>
        <td>年龄：</td>
        <td>
        <%if(age==null||"".equals(age)){
                out.println("");
        }else{
                out.println(age);
        } %>
        </td>
    </tr>
    <tr>
        <td>Email：  </td>
        <td>
        <%if(email==null||"".equals(email)){
                out.println("");
        }else{
                out.println(email);
        } %>
        </td>
    </tr>
    <tr><td align="center" colspan="4" height="20"></td></tr>
</table>
```

秘笈心法

在获取表单请求参数时，很容易发生乱码问题，这时，需要在应用 request 获取请求参数之前，调用 request

对象的 setCharacterEncoding()方法设置请求编码。需要注意的是，这个编码是根据表单页的编码格式而定的，如表单页的编码格式为 UTF-8，那么在处理页中应该调用如下代码：

```
request.setCharacterEncoding("UTF-8");
```

实例 103	获取访问请求参数 光盘位置：光盘\MR\04\103	初级 实用指数：★★

实例说明

在 Web 应用程序中，有时需要通过超链接来传递参数。假定在显示员工信息的页面中，每行员工信息都包含一个“删除”的超链接，然后在这个超链接中需要指定用户 ID 为参数，当单击“删除”超链接时，在处理页中需要获取用户 ID 值，最后根据用户 ID 删除员工信息。本实例将介绍如何获取超链接访问的请求参数，实例运行结果如图 4.11 所示。

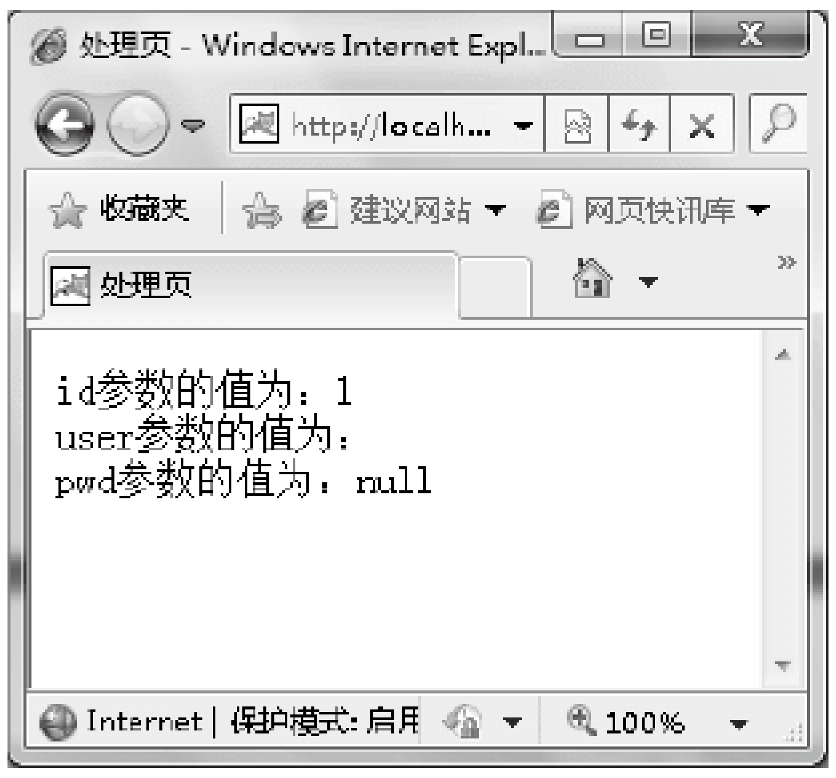


图 4.11　处理页的运行结果

关键技术

request 对象用于处理 HTTP 请求中的各项参数。在这些参数中，最常用的就是获取访问请求参数。当通过超链接的形式发送请求时，可以为该请求传递参数，这可以通过在超链接的后面加上问号?来实现。注意这个问号为英文半角的符号。例如，发送一个请求到 delete.jsp 页面，并传递一个名称为 id 的参数，可以通过以下超链接实现：

```
<a href="delete.jsp?id=1">删除</a>
```

设计过程

（1）创建 index.jsp 文件，在该文件中添加一个用于链接到 deal.jsp 页面的超链接，并传递两个参数。index.jsp 文件的具体代码如下：

```
<%@ page language="java" contentType="text/html; charset=GB18030" pageEncoding="GB18030"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=GB18030">
<title>使用 request 对象获取请求参数值</title>
</head>
<body>
<a href="deal.jsp?id=1&user=">处理页</a>
</body>
</html>
```

（2）创建 deal.jsp 文件，在该文件中通过 request 对象的 getParameter()方法获取请求参数 id、user 和 pwd 的值并输出。deal.jsp 文件的具体代码如下：

```
<%@ page language="java" contentType="text/html; charset=GB18030" pageEncoding="GB18030"%>
<%
 String id = request.getParameter("id");              //获取 id 参数的值
 String user = request.getParameter("user");          //获取 user 参数的值
 String pwd = request.getParameter("pwd");            //获取 pwd 参数的值
%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=GB18030">
<title>处理页</title>
</head>
<body>
id 参数的值为：<%=id%><br>
user 参数的值为：<%=user%><br>
pwd 参数的值为：<%=pwd%>
```

```
</body>
</html>
```

秘笈心法

在通过问号“?”来指定请求参数时，参数值不需要使用单引号或双引号括起来，包括字符型的参数。可以同时指定多个参数，各参数间用符号“&”分隔即可。

实例 104 将表单请求提交到本页

光盘位置：光盘\MR\04\104

初级

实用指数：★★

实例说明

在 Web 应用程序中，有时需要将表单请求提交到本页进行处理。本实例将介绍如何将表单请求提交到本页。运行本实例，如图 4.12 所示，输入用户名和密码，单击“登录”按钮，将在本页直接获取到用户名和密码，并显示在页面中。



图 4.12　将表单请求提交到本页

关键技术

实现将表单提交到本页，实际上很简单，只要将表单<form>的 action 属性设置为本页即可，假定表单页为 index.jsp，那么 action 的值就应该为 index.jsp，然后同样应用 request 对象的 getParameter()方法来获取表单元素的值。

设计过程

（1）创建表单页 index.jsp，并设置将表单信息提交到本页，关键代码如下：

```
<form action="index.jsp" method="post">
    <table height="150" width="300">
        <tr><td align="center" colspan="4" height="20"></td></tr>
            <tr><td align="center" colspan="4">用户登录</td></tr>
            <tr>
                <td>用户名：  </td>
                <td><input type="text" name="name" /></td>
            </tr>
            <tr>
                <td>密码：  </td>
                <td><input type="password" name="pwd" /></td>
            </tr>
            <tr>
                <td></td>
                <td><input type="submit" value="登  录" /></td>
            </tr>
            <tr>
                <td>用户名参数为：</td>
                <td><%=name %></td>
            </tr>
            <tr>
                <td>密码参数为：</td>
                <td><%=pwd %></td>
            </tr>
            <tr><td align="center" colspan="4" height="20"></td></tr>
        </table>
    </form>
```

（2）在 index.jsp 页中获取表单信息，关键代码如下：

```
<%
    request.setCharacterEncoding("UTF-8");
    String name = request.getParameter("name");
    String pwd = request.getParameter("pwd");
%>
```

秘笈心法

需要注意的是，在本实例的页面中使用 request 对象的 getParameter()方法来获取参数时，在页面被加载后就会自动执行这段代码，因此，在本实例中需要进行合理的判断，只有提交表单后才可以执行 request 对象的 getParameter()方法来获取表单的代码。针对这一特点，可以在表单的“提交”按钮中添加一个 name 属性，以本实例为例，在“登录”按钮的标签中添加 name 属性，代码如下：

```
<input type="submit" name="submit" value="注册">
```

然后在获取表单的请求信息之前，判断用户是否单击了“登录”按钮，只有“登录”按钮的值不为 null 时，才继续执行，代码如下：

```
<%
if( request.getParameter("submit")!=null ){
        request.setCharacterEncoding("UTF-8");
        String name = request.getParameter("name");
        String pwd = request.getParameter("pwd");
}
%>
```

实例 105　通过 request 对象进行数据传递

光盘位置：光盘\MR\04\105

初级

实用指数：★★

实例说明

在进行请求转发时，需要把一些数据传递到转发后的页面进行处理。这时，就需要使用 request 对象的 setAttribute()方法将数据保存到 request 范围内的变量中。本实例将介绍如何应用 request 对象进行数据传递，运行本实例，将显示如图 4.13 所示的运行结果。

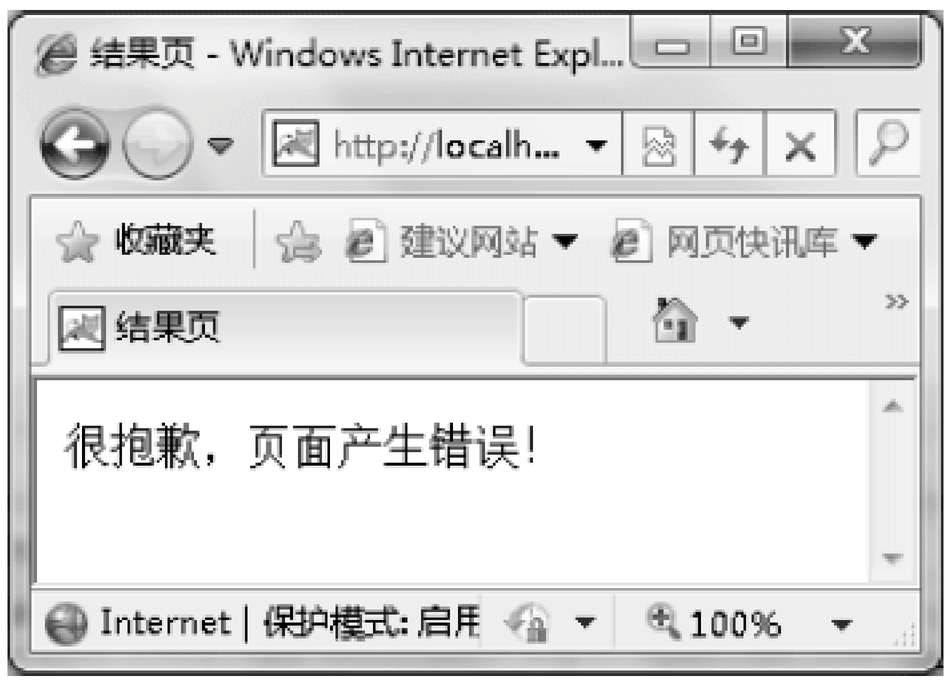


图 4.13　本实例的运行结果

关键技术

request 对象可以视为一个域，可以应用 setAttribute()方法向域范围内存放数据。request 对象的 setAttribute()方法的语法格式如下：

```
request.setAttribute(String name,Object object);
```

参数说明

❶ name：表示变量名，为 String 类型，在转发后的页面取数据时，就是通过这个变量名来获取数据的。

❷ object：用于指定需要在 request 范围内传递的数据，为 Object 类型。

在将数据保存到 request 范围内的变量中后，可以通过 request 对象的 getAttribute()方法获取该变量的值，具体的语法格式如下：

```
request.getAttribute(String name);
```

参数说明

name：表示变量名，该变量名在 request 范围内有效。

设计过程

（1）创建 index.jsp 文件，在该文件中，首先应用 Java 的 try…catch 语句捕获页面中的异常信息，如果没有异常，则将运行结果保存到 request 范围内的变量中；如果出现异常，则将错误提示信息保存到 request 范围内的变量中，然后应用<jsp:forward>动作指令将页面转发到 deal.jsp 页面。index.jsp 文件的具体代码如下：

```
<%@ page language="java" contentType="text/html; charset=GB18030" pageEncoding="GB18030"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=GB18030">
<title>Insert title here</title>
</head>
<body>
<%
try{                                                    //捕获异常信息
        int money=100;
        int number=0;
        request.setAttribute("result",money/number);    //保存执行结果
}catch(Exception e){
        request.setAttribute("result","很抱歉，页面产生错误！");   //保存错误提示信息
}
%>
<jsp:forward page="deal.jsp"/>
</body>
</html>
```

（2）创建 deal.jsp 文件，在该文件中通过 request 对象的 getAttribute()方法获取保存在 request 范围内的变量 result 并输出。这里需要注意的是，由于 getAttribute()方法的返回值为 Object 类型，所以需要调用其 toString()方法将其转换为字符串类型。deal.jsp 文件的具体代码如下：

```
<%@ page language="java" contentType="text/html; charset=GB18030"
    pageEncoding="GB18030"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=GB18030">
<title>结果页</title>
</head>
<body>
<%String message=request.getAttribute("result").toString(); %>
<%=message %>
</body>
</html>
```

秘笈心法

由于应用 setAttribute()方法保存在 request 对象中的数据类型为 Object，因此，在实现一些数据的传递时，可以将类的对象保存在 request 中。假定有一个保存用户注册信息的 User 类对象，可以将这个对象保存在 request 中进行传递，然后在获取 User 对象时需要进行类型强制转换，将 Object 类型转换为 User 类型，代码如下：

```
User user = (User)request.getAttribute("user");
```

实例 106 通过 cookie 保存并读取用户登录信息

初级

光盘位置：光盘\MR\04\106

实用指数：★★

实例说明

在互联网中，cookie 是小段的文本信息，在网络服务器上生成，并发送给浏览器。通过使用 cookie 可以标识用户身份，记录用户名和密码，跟踪重复用户等。浏览器将 cookie 以 key/value 的形式保存到客户机的某个指定目录中。运行本实例，第一次显示的页面如图 4.14 所示；输入姓名 mr，并单击“确定”按钮后，将显示如图 4.15 所示的运行结果。



图 4.14　第一次运行的结果



图 4.15　第二次运行的结果

关键技术

通过 cookie 的 getCookies()方法可获取所有 cookie 对象的集合，通过 cookie 对象的 getName()方法可获取指定名称的 cookie，通过 getValue()方法可获取 cookie 对象的值。另外，将一个 cookie 对象发送到客户端使用了 response 对象的 addCookie()方法。

设计过程

（1）创建 index.jsp 文件，在该文件中首先获取 cookie 对象的集合，如果集合不为空，就通过 for 循环遍历 cookie 集合，从中找出设置的 cookie（这里设置为 mrCookie），并从该 cookie 中提取出用户名和注册时间，再根据获取的结果显示不同的提示信息。index.jsp 文件的具体代码如下：

```
<%@ page language="java" contentType="text/html; charset=GB18030"
 pageEncoding="GB18030"%>
<%@ page import="java.net.URLDecoder" %>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=GB18030">
<title>通过 cookie 保存并读取用户登录信息</title>
</head>
<body>
<%
 Cookie[] cookies = request.getCookies();                          //从 request 中获得 Cookie 对象的集合
 String user = "";                                                 //登录用户
 String date = "";                                                 //注册的时间
 if (cookies != null) {
        for (int i = 0; i < cookies.length; i++) {                 //遍历 cookie 对象的集合
               if (cookies[i].getName().equals("mrCookie")) {      //如果 cookie 对象的名称为 mrCookie
                      user = URLDecoder.decode(cookies[i].getValue().split("#")[0]);  //获取用户名
                      date = cookies[i].getValue().split("#")[1];  //获取注册时间
               }
        }
 }
 if ("".equals(user) && "".equals(date)) {                         //如果没有注册
%>
        游客您好，欢迎您初次光临！
        <form action="deal.jsp" method="post">
               请输入姓名：<input name="user" type="text" value="">
               <input type="submit" value="确定">
        </form>
<%
 } else {                                                          //已经注册
%>
        欢迎[<b><%=user %></b>]再次光临<br>
        您注册的时间是：<%=date %>
<%
 }
%>
</body>
</html>
```

（2）编写 deal.jsp 文件，用于向 cookie 中写入注册信息。deal.jsp 文件的具体代码如下：

```
<%@ page language="java" contentType="text/html; charset=GB18030" pageEncoding="GB18030"%>
<%@ page import="java.net.URLEncoder" %>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=GB18030">
<title>写入 cookie</title>
</head>
<body>
<%
request.setCharacterEncoding("GB18030");                                    //设置请求的编译为 GB18030
String user=URLEncoder.encode(request.getParameter("user"),"GB18030");      //获取用户名
Cookie cookie = new Cookie("mrCookie", user+"#"+new java.util.Date().toLocaleString());//创建并实例化 cookie 对象
cookie.setMaxAge(60*60*24*30);                                              //设置 cookie 有效期 30 天
response.addCookie(cookie);                                                 //保存 cookie
%>
<script type="text/javascript">window.location.href="index.jsp"</script>
</body>
</html>
```

秘笈心法

在向 cookie 保存的信息中，如果包括中文，则需要调用 java.net.URLEncoder 类的 encode()方法将要保存到 cookie 中的信息进行编码；在读取 cookie 的内容时，还需要应用 java.net.URLDecoder 类的 decode()方法进行解码。这样，就可以成功地向 cookie 中写入中文信息。

实例 107	实现重定向页面 光盘位置：光盘\MR\04\107	初级 实用指数：★★

实例说明

运行本实例，默认执行的是 index.jsp 页面，在该页面中，又执行了重定向页面到 login.jsp 的操作，所以在浏览器中将显示如图 4.16 所示的用户登录页面。

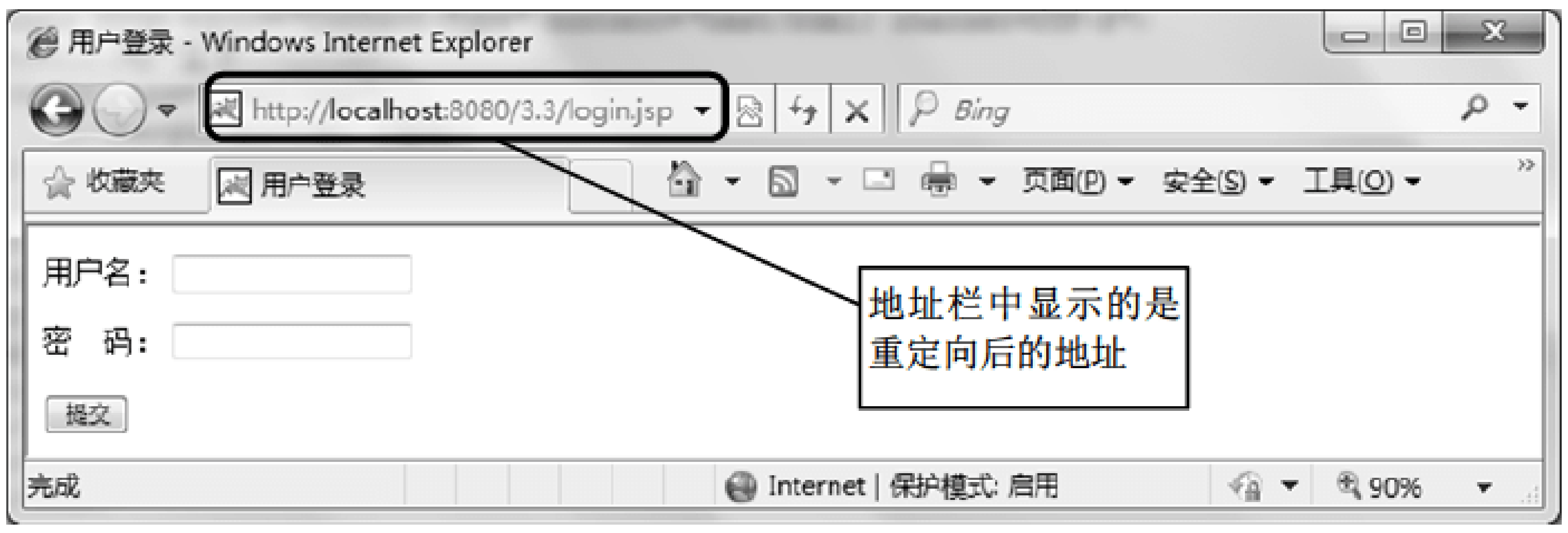


图 4.16　实现重定向页面

关键技术

使用 response 对象提供的 sendRedirect()方法可以将网页重定向到另一个页面。sendRedirect()方法的语法格式如下：

```
response.sendRedirect(String path);
```

参数说明

path：用于指定目标路径，可以是相对路径，也可以是不同主机的其他 URL 地址。

例如，使用 sendRedirect()方法重定向网页到 login.jsp 页面（与当前网页同级）和明日编程词典网（与该网页不在同一主机）的代码如下：

```
response.sendRedirect("login.jsp");            //重定向到 login.jsp 页面
response.sendRedirect("www.mrbccd.com");       //重定向到明日编程词典网
```

设计过程

（1）创建 index.jsp 文件，在该文件中调用 response 对象的 sendRedirect()方法重定向页面到用户登录页面 login.jsp。index.jsp 文件的关键代码如下：

```
<%@ page language="java" contentType="text/html; charset=GB18030" pageEncoding="GB18030"%>
<%response.sendRedirect("login.jsp"); %>
```

（2）编写 login.jsp 文件，在该文件中添加用于收集用户登录信息的表单及表单元素，关键代码如下：

```
<form name="form1" method="post" action="">
用户名：  <input    name="name" type="text" id="name" style="width: 120px"><br>
密  码：  <input name="pwd" type="password" id="pwd" style="width: 120px"> <br>
<br>
<input type="submit" name="Submit" value="提交">
</form>
```

秘笈心法

重定向操作支持将地址重定向到不同的主机上，这一点与转发不同。在客户端浏览器上将会得到跳转的地址，并重新发送请求链接。用户可以从浏览器的地址栏中看到跳转后的地址。进行重定向操作后，request 中的属性全部失效，并且开始一个新的 request 对象。

实例 108　防止表单在网站外部提交

光盘位置：光盘\MR\04\108

初级　实用指数：★★

实例说明

如果静态网页中含有用户提交的表单和字段信息，而从网页的源代码中又可以看到网页被提交的目标地址，那么修改静态页面表单提交的目标地址，就可以实现在本地运行静态网页并向服务器提交数据的功能。这样，任何人都可以利用网页在网站外登录网站，从而给网站留下严重的安全隐患。为解决该问题，本实例介绍一种防止表单在网站外部提交的方法，运行本实例，如图 4.17 所示，在“用户名”文本框中输入用户名，在“密码”文本框中输入密码，单击“提交”按钮即可进入到网页的处理页面，此时地址栏中的地址即为处理页地址（用户也可以通过其他方法获得），当用户在本地计算机上编写静态表单页时，将目标地址设置为以上地址后，运行网页并提交表单将显示如图 4.18 所示的提示信息。



图 4.17　表单提交前



图 4.18　从外部提交表单后

关键技术

本实例首先使用 request.getRequestURL().toString()方法获得当前网页的 IE 地址，然后使用 request.getHeader ("referer")请求页面的地址，接着使用 URL urlOne = new URL(String url)的方法分别获取两个 IE 地址服务器主机名，最后比较两个主机的名称是否相同。如果网页的 URL 为空，比较时将出现路径有误的信息。

设计过程

（1）创建 index.jsp 页面，添加表单及相关的表单元素，并设置 Form 表单的相关属性值，关键代码如下：

```
<form name="form1" action="doform.jsp" method="post">
            <table align="center">
```

```
            <tr>
                <td> </td>
            </tr>
            <tr>
                <td>用户名：</td>
                <td><input type="text" name="name"></td>
            </tr>
            <tr>
                <td>密 码:</td>
                <td><input type="password" name="pass"></td>
            </tr>
           <tr>
                 <td align="center" colspan="2">
                     <input type="submit" name="action2" value="提交">
                     <input type="reset" name="Submit" value="重置">
                </td>
            </tr>
        </table>
</form>
```

（2）如果表单被提交，将判断表单提交的路径是否有误，如果提交的路径有误，系统将给予提示，并禁止从网站外部进行提交表单，关键代码如下：

```
<%
        String address = request.getHeader("referer");           //获取页面的请求地址
        String pathAdd = "";                                     //定义空字符串
        if (address != null) {                                   //判断当页面的请求地址为空时
                URL urlOne = new URL(address);                   //实例化 URL 方法
                pathAdd = urlOne.getHost();                      //获取请求页面的服务器主机
        }
        String address1 = request.getRequestURL().toString();  //获取当前网页的地址
        String pathAdd1 = "";
        if (address1 != null) {
                URL urlTwo = new URL(address1);
                pathAdd1 = urlTwo.getHost();                     //获取当前网页的服务器主机
        }
%>
<body>
        <table align="center">
        <tr><td>  </td></tr>
                <tr>
                        <td>
                                <%
                                        if (!pathAdd.equals(pathAdd1)) { //判断当前页面的主机与服务器的主机是否相同
%>
```

秘笈心法

由于网站服务器的名称是唯一的，而且每次从客户端浏览服务器网页时，所浏览的网页中包含了网页的来源信息，因此可以通过比较服务器名称的方法来防止表单在网站外被提交。

实例 109　通过 Application 对象实现网站计数器

光盘位置：光盘\MR\04\109

初级

实用指数：★★

实例说明

Application 对象是 JSP 的一个内建对象。当服务器启动时，该对象会被自动创建，直到服务器关闭该对象才会消失，并且在此期间可以被多个用户共同使用。这是不同于 session 对象的。本实例将使用 Application 对象实现网站计数器，程序运行结果如图 4.19 所示。

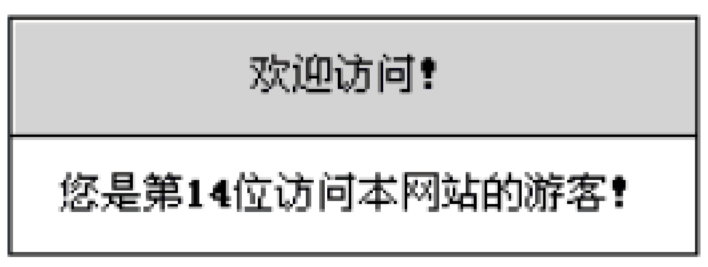


图 4.19　Application 对象实现的网站计数器

关键技术

本实例主要使用 Application 对象的 setAttribute()和 getAttribute()方法，下面分别进行介绍。

（1）setAttribute()方法

语法格式如下：

```
setAttribute(String key,String value)
```

功能：在 Application 对象中添加属性。

参数说明

❶ key：此参数设置属性名称。

❷ value：此参数设置属性值。

（2）getAttribute()方法

语法格式如下：

```
getAttribute(String key)
```

功能：获取指定属性的值。

参数说明

key：此参数意义同 setAttribute()方法的 key 参数。

设计过程

创建一个首页面 index.jsp，当用户访问该页面时，程序会自动将网站的访问次数加 1，关键代码如下：

```
<%
 int i=0;
 synchronized(application){
      if(application.getAttribute("times")==null){                    //服务器启动后的第一位访问者
          i=1;
      }
      else{
           i=Integer.parseInt((String)application.getAttribute("times"));
           i++;                                                        //访问次数加 1
      }
      application.setAttribute("times",Integer.toString(i));          //将访问次数存入 Application 对象中
 }
%>
      <table>
          <tr bgcolor="lightgrey">
            <td align="center">欢迎访问！</td>
          </tr>
          <tr>
            <td align="center">
              您是第<b><%=i%></b>位访问本网站的游客！
            </td>
          </tr>
        </table>
```

秘笈心法

synchronized(Object){…}方法使多用户同步共享同一个对象。Object 表示被共享的对象，{}中包含对该对象进行操作的代码。当被共享的对象正在被使用时，其他的用户只能等待，直到上一个用户对其操作结束。

实例 110	记录用户 IP 地址的计数器 光盘位置：光盘\MR\04\110	初级 实用指数：★★

实例说明

Application 对象是 JSP 的一个内置对象，当服务器启动时自动创建直到服务器关闭。本实例是用 Application

对象的 setAttribute()和 getAttribute()方法来实现网站计数器功能，运行结果如图 4.20 所示。

第N位访问者	访问者IP地址	访问时间
45	192.168.1.128	2006-10-9 11:12:38
46	192.168.1.64	2006-10-10 02:12:42
47	192.168.1.57	2006-10-11 03:12:44
48	192.168.1.57	2006-10-11 04:37:35
49	127.0.0.1	2006-10-12 05:35:31
您是第49为访问者！ 您的IP为：127.0.0.1 您访问的时间为：2006-10-12 03:35:31		

图 4.20　记录用户 IP 地址和时间的计数器

关键技术

本实例应用了 request 对象的 getRemoteAddr()方法，其语法格式如下：

```
getRemoteAddr()
```

功能：获得用户的 IP 地址，返回一个字符串型值。

设计过程

（1）创建一个对数据库操作的 DB 类，关键代码如下：

```
package com.count.Online;
import java.sql.*;
public class DB {
    private Connection con;
    private Statement stm;
    private ResultSet rs;
    private String classname="com.microsoft.jdbc.sqlserver.SQLServerDriver";
    private String url="jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=db_database07";
    public DB(){}
    public Connection getCon(){
        try{
            Class.forName(classname);
        }catch(ClassNotFoundException e){
            e.printStackTrace();
        }
        try{
            con=DriverManager.getConnection(url,"sa","");
        }catch(Exception e){
            e.printStackTrace(System.err);
            con=null;
        }
        return con;
    }
    public Statement getStmed(){
        try{
           con=getCon();
           stm=con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                                  ResultSet.CONCUR_READ_ONLY);
        }catch(Exception e){e.printStackTrace(System.err);}
        return stm;
    }
    public ResultSet search(String sql){
        getStmed();
        try{
            rs=stm.executeQuery(sql);
        }catch(Exception e){e.printStackTrace();}
        return rs;
    }
    public int dosql(String sql){
        int i=-1;
        getStmed();
        try{
            i=stm.executeUpdate(sql);
        }catch(Exception e){e.printStackTrace();}
```

```
        return i;
    }
}
```

（2）创建一个 CountOnline 类，在该类中调用 DB 类中的 dosql()方法向数据库中增加记录，关键代码如下：

```
package com.count.Online;
import java.sql.*;
public class CountOnline {
    private String userip;                                                    //用户 IP 地址
    private String nowdate;                                                   //用户访问时间
    private int times;                                                        //网站的访问次数
    private DB db=new DB();
    public CountOnline(){}
    …//省略了属性的 setXXX()和 getXXX()方法
    public ResultSet adduser(){
        ResultSet rs=null;
        String sql="insert into tb_IPcount values("+this.times+",'"+this.userip+"','"+this.nowdate+"')";    //生成 SQL 语句
        try{
            db.dosql(sql);                                                    //执行 SQL 语句
            rs=db.search("select * from tb_IPcount");                         //查询表中的记录
        }catch(Exception e){e.printStackTrace();}
        return rs;                                                            //返回结果集
    }
    public void dbclose(){
        db.closed();                                                          //释放资源
    }
}
```

（3）创建一个供用户访问的页面 index.jsp。在该页面中获得网站的访问次数、访问用户的 IP 地址和访问时间，然后调用 CountOnline 类中的方法进行操作，关键代码如下：

```
<%@ page import="java.util.Date,java.text.*,java.sql.*" %>
<jsp:useBean id="mycount" class="com.count.Online.CountOnline"/>
<jsp:useBean id="mydb" class="com.count.Online.DB"/>
<%
   SimpleDateFormat format=new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
   String sql="select MAX(user_order) from tb_IPcount as max";                //查询最后一次登录的次数
   ResultSet rs=mydb.search(sql);
   rs.next();
   int max=rs.getInt(1);
   mydb.closed();
   mycount.setTimes(max+1);                                                   //设置用户的访问排名
   String ip=request.getRemoteAddr();
   mycount.setUserip(ip);                                                     //设置用户 IP 地址
   String nowdate=format.format(new Date());
   mycount.setNowdate(nowdate);                                               //设置用户的访问时间
   rs=mycount.adduser();                                                      //增加该用户的访问信息
%>
     <table>
       <tr bgcolor="lightgrey">
        <td align="center">第 N 位访问者</td>
        <td align="center">访问者 IP 地址</td>
        <td align="center">访问时间</td>
       </tr>
       <%
          while(rs.next()){
       %>
       <tr>
         <td align="center"><%=rs.getInt("user_order")%></td>
         <td align="center"><%=rs.getString("user_ip")%></td>
         <td align="center"><%=rs.getString("user_time")%></td>
       </tr>
       <%
          }
          mycount.dbclose();                                                  //关闭数据库连接
       %>
       <tr>
        <td align="center" colspan="3">
          您是第<%=max+1%>位访问者！
```

```
            <br>
            您的 IP 为：<%=ip%>
            <br>
            您访问的时间为：<%=nowdate%>
        </td>
      </tr>
    </table>
```

秘笈心法

在使用 Application 全局变量时需要注意的是，它是被多个用户共享的，所以当 Application 对象正在被使用时应避免其他用户再对其进行操作。

实例 111 只对新用户计数的计数器

初级

光盘位置：光盘\MR\04\111

实用指数：★★★

实例说明

通过本实例可以计算出有多少个 IP 访问网站，也可以计算出每个 IP 地址的访问次数。运行本实例，程序将记录用户的 IP 地址和访问次数，运行结果如图 4.21 所示。

访问者IP地址	访问次数qq
127.0.0.1	22
192.168.0.13	45
192.168.0.14	78
192.168.0.17	124
192.168.0.54	154
您的IP为：127.0.0.1 您的访问次数为：22次 共有 5 个新用户访问过本页	

图 4.21 只对新用户计数的计数器

关键技术

得到用户 IP 地址后，首先在数据表中查询该 IP 地址，如果存在则更新该 IP 地址的访问次数，否则插入一条新的记录并将访问次数设为 1。

本实例中使用 getRemoteAddr()方法获取用户的 IP 地址，其语法格式如下：

```
request.getRemoteAddr()
```

功能：获取用户的 IP 地址，返回值为字符型。

其中参数 request 为 JSP 的内建对象。

获取访问过的人数实际上就是数据表中的记录总数，这里可以使用 ResultSet 类的 getRow()方法获得，其语法格式如下：

```
rs.getRow()
```

功能：获取当前记录的排列行号，返回 int 类型值。

其中参数 rs 为 ResultSet 类对象。

设计过程

（1）创建一个对数据库操作的 DB 类，可参见实例 110 或光盘中的相关代码。

（2）创建 CountOnline 类，在该类中调用 DB 类中的 search()和 dosql()方法对数据库进行操作，关键代码如下：

```
package com.count.Online;
import java.sql.*;
public class CountOnline {
    private String userip;
    private String nowdate;
    private int times;
    private DB db=new DB();
    public CountOnline(){}
    …//省略了属性的 setXXX()和 getXXX()方法
    public ResultSet checkuser(){
        String sql="select * from tb_newusercount where user_ip='"+this.userip+"'";
        ResultSet rs=null;
        try{
```

```
            rs=db.search(sql);
            if(rs.next()){    //如果访问者的 IP 存在于数据表中，则更新访问次数
              this.times=rs.getInt("user_times")+1;
              sql="update tb_newusercount set user_times="+this.times+" where user_ip='"+this.userip+"'";
              db.dosql(sql);
            }else{               //否则插入新的记录
                this.times=1;
                sql="insert into tb_newusercount values('"+this.userip+"',1)";
                db.dosql(sql);
            }
            rs=db.search("select * from tb_newusercount");    //返回所有记录
        }catch(Exception e){e.printStackTrace();}
        return rs;
    }
    public void dbclose(){
        db.closed();
    }
}
```

（3）创建首页 index.jsp。在该页面中获得访问用户的 IP 地址，然后调用 CountOnline 类中的方法进行操作，关键代码如下：

```
<%@ page import="java.sql.*" %>
<jsp:useBean id="mycount" class="com.count.Online.CountOnline"/>
<%
  String ip=request.getRemoteAddr();
  mycount.setUserip(ip);
  ResultSet rs=mycount.checkuser();
  rs.last();
  int num=rs.getRow();
%>
      <table>
        <tr bgcolor="lightgrey">
          <td align="center">访问者 IP 地址</td>
          <td align="center">访问次数</td>
        </tr>
        <%
          rs.beforeFirst()
          while(rs.next()){
        %>
        <tr>
            <td align="center"><%=rs.getString("user_ip")%></td>
            <td align="center"><%= rs.getInt("user_times")%></td>
        </tr>
        <%
          }
        %>
        <tr>
          <td align="center" colspan="2">
            您的 IP 为：<%=ip%>
            <br>
            您的访问次数为：<%=mycount.getTimes()%>次
            <br>
            共有 <%=num%> 个新用户访问过本页
          </td>
        </tr>
       </table>
       <%
          mycount.dbclose();
       %>
```

秘笈心法

在使用 ResultSet 类的 getRow()方法获取总记录数之前，应使用 ResultSet 类的 last()方法将记录指针指向最后一条记录，其语法格式如下：

```
rs.last()
```

功能：将记录指针指向最后一条记录，返回 boolean 类型值。

其中参数 rs 为 ResultSet 类对象。

ResultSet 类能够使用 last()和 getRow()方法的前提是，将查询结果集设为可滚动的。

实例 112 统计用户在某一页停留的时间

光盘位置：光盘\MR\04\112

初级

实用指数：★★

实例说明

运行本实例后将会显示用户的登录时间，并且每过 10 秒刷新一次页面显示用户的停留时间，实例运行结果如图 4.22 所示。



图 4.22　统计用户在页面的停留时间

关键技术

当用户访问页面时记录用户的访问时间，然后求出当前时间与用户登录时间的时间差，即为用户停留时间。将 session 的有效活动时间设置为稍大于页面刷新的时间。

本实例使用的函数和方法分别介绍如下。

（1）Date 类的构造函数

语法格式如下：

```
Date time=new Date()
```

功能：获取当前时间。

（2）Date 类的 getTime()方法

语法格式如下：

```
DateObject.getTime()
```

功能：获取从 1970 年 1 月 1 日午夜起的毫秒数，返回 long 类型的数值。

其中参数 DateObject 为 Date 类对象。

（3）Date 类的 toLocalString()方法

语法格式如下：

```
DateObject.toLocalString()
```

功能：将时间格式化为本地时间格式，返回值为字符型。

其中参数 DateObject 为 Date 类对象。

（4）JSP 内置对象 session 的 setMaxInactiveInterval()方法

语法格式如下：

```
session.setMaxInactiveInterval(int num)
```

功能：设置 session 对象的有效活动时间，该时间以秒为单位。

其中参数 num 表示设置的秒数。

（5）JSP 内置对象 session 的 isNew()方法

语法格式如下：

```
session.isNew()
```

功能：判断当前用户是否是新用户，返回 boolean 类型值。

当用户刷新网页时通过此方法得到的值为 false。

设计过程

（1）创建 StopTime.java 类来计算用户停留的时间，关键代码如下：

```
package com.count.stoptime;
import java.util.*;
public class StopTime {
    private int h=0;
```

```
    private int m=0;
    private int s=0;
    public StopTime(){}
    public void counttime(Date start){
        Date end=new Date();                                //获得当前时间
        long howmuch=end.getTime()-start.getTime();         //获得当前时间与登录时间相差的毫秒数
        h=(int)(howmuch/1000/60/60);                        //计算小时
        howmuch=howmuch-h*60*60*1000;
        m=(int)(howmuch/1000/60);                           //计算分钟
        howmuch=howmuch-m*60*1000;
        s=(int)(howmuch/1000);                              //计算停留的秒数
    }
    …//省略了属性的 getXXX()方法
}
```

（2）创建首页面 index.jsp，用于显示用户的停留时间，主要代码如下：

```
<%@ page import="java.util.*" %>
<jsp:useBean id="mycounttime" class="com.count.stoptime.StopTime" scope="page"/>
<%
    session.setMaxInactiveInterval(11);                 //设置 session 的有效活动时间为 11s
    Date now=new Date();                                //获取当前时间
    if(session.isNew()){                                //如果是新用户，记录用户登录时间
      session.setAttribute("start",now);
    }
    else{                                               //调用以用户登录时间为参数的 counttime()方法，计算停留时间
        mycounttime.counttime((Date)session.getAttribute("start"));
    }
%>
    <meta http-equiv="refresh" content="10"> <!--设置刷新时间-->
    <table>
      <tr>
        <td align="center">
          您登录的时间为：<%=((Date)session.getAttribute("start")).toLocaleString()%>
        </td>
      </tr>
      <tr>
        <td align="center">
          您在本页的停留时间为：<%=mycounttime.getH()%>小时<%=mycounttime.getM()%>分
                                                    <%=mycounttime.getS()%>秒
        </td>
      </tr>
    </table>
```

秘笈心法

设置页面的刷新时间还可以使用另一种方法——response.setHeader("Refresh","10")，该方法表示 10 秒刷新一次页面。

实例 113	应用 session 对象实现用户登录 光盘位置：光盘\MR\04\113	初级 实用指数：★★

实例说明

session 在网络中被称为会话。由于 HTTP 协议是一种无状态协议，也就是当一个客户向服务器发出请求时，服务器接收请求并返回响应后，该连接结束，而服务器并不保存相关的信息。为了弥补这一缺点，HTTP 协议提供了 session。通过 session 可以在应用程序的 Web 页间进行跳转，并保存用户的状态，使整个用户会话一直存在下去，直到关闭浏览器。但是，如果在一个会话中，客户端长时间不向服务器发出请求，session 对象就会自动消失。这个时间取决于服务器，例如，Tomcat 服务器默认为 30 分钟。不过这个时间可以通过编写程序进行修改。使用 session 对象一个最常用的功能就是记录用户的状态。下面将通过一个具体的实例介绍应用 session 对象实现用户登录。运行本实例，首先进入的是用户登录页面，输入用户名（mr）和密码（mrsoft）后，单击

“提交”按钮，将显示如图 4.23 所示的系统主页，如果输入用户名 mr，密码不输入 mrsoft，则重新返回到用户登录页面。在系统主页，单击“退出”超链接，将销毁当前 session，重新返回到用户登录页面。

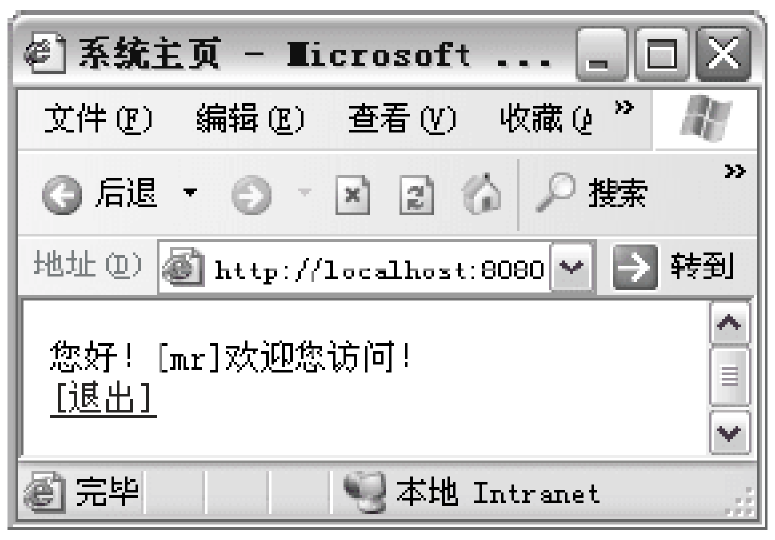


图 4.23　系统主页

关键技术

通过 session 对象可以存储或读取客户相关的信息，如用户名、购物信息等。这可以通过 session 对象的 setAttribute()方法和 getAttribute()方法实现，下面分别进行介绍。

（1）setAttribute()方法

该方法用于将信息保存在 session 范围内，语法格式如下：

```
session.setAttribute(String name,Object obj)
```

参数说明

❶ name：用于指定作用域在 session 范围内的变量名。

❷ obj：保存在 session 范围内的对象。

例如，将用户名“无语”保存到 session 范围内的 username 变量中，可以使用下面的代码：

```
session.setAttribute("username","无语");
```

（2）getAttribute()方法

该方法用于获取保存在 session 范围内的信息，语法格式如下：

```
getAtttibute(String name)
```

参数说明

name：指定保存在 session 范围内的关键字。

例如，读取保存到 session 范围内的 username 变量的值，可以使用下面的代码：

```
session.getAttribute("username");
```

设计过程

（1）创建 index.jsp 文件，在该文件中添加用于收集用户登录信息的表单及表单元素，关键代码如下：

```
<form name="form1" method="post" action="">
用户名： <input    name="name" type="text" id="name" style="width: 120px"><br>
密  码： <input name="pwd" type="password" id="pwd" style="width: 120px"> <br>
<br>
<input type="submit" name="Submit" value="提交">
</form>
```

（2）编写 deal.jsp 文件，在该文件中模拟用户登录（这里将用户信息保存到一个二维数组中），如果用户登录成功，将用户名保存到 session 范围内的变量中，并将页面重定向到 main.jsp 页面；否则，将页面重定向到 index.jsp 页面，重新登录。deal.jsp 文件的具体代码如下：

```
<%@ page language="java" contentType="text/html; charset=GB18030" pageEncoding="GB18030"%>
<%@ page import="java.util.*" %>
<%
String[][] userList={{"mr","mrsoft"},{"wgh","111"},{"sk","111"}};          //定义一个保存用户列表的二维数组
boolean flag=false;                                                        //登录状态
request.setCharacterEncoding("GB18030");                                   //设置编码
String username=request.getParameter("username");                          //获取用户名
String pwd=request.getParameter("pwd");                                    //获取密码
for(int i=0;i<userList.length;i++){                                        //遍历二维数组
 if(userList[i][0].equals(username)){                                      //判断用户名
        if(userList[i][1].equals(pwd)){                                    //判断密码
                flag=true;                                                 //表示登录成功
```

```
            break;                                          //跳出 for 循环
        }
    }
}
if(flag){                                                   //如果值为 true，表示登录成功
 session.setAttribute("username",username);                 //保存用户名到 session 范围的变量中
 response.sendRedirect("main.jsp");                         //跳转到主页
}else{
 response.sendRedirect("index.jsp");                        //跳转到用户登录页面
}
%>
```

（3）编写 main.jsp 文件，在该文件中，首先获取并显示保存到 session 范围内的变量，然后添加一个退出超链接。main.jsp 文件的具体代码如下：

```
<%@ page language="java" contentType="text/html; charset=GB18030" pageEncoding="GB18030"%>
<%
String username=(String)session.getAttribute("username");     //获取保存在 session 范围内的用户名
%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=GB18030">
<title>系统主页</title>
</head>
<body>
您好！[<%=username %>]欢迎您访问！<br>
<a href="exit.jsp">[退出]</a>
</body>
</html>
```

（4）编写 exit.jsp 文件，在该文件中销毁 session，并重定向页面到 index.jsp 页面。exit.jsp 文件的具体代码如下：

```
<%@ page language="java" contentType="text/html; charset=GB18030" pageEncoding="GB18030"%>
<%
session.invalidate();                                       //销毁 session
response.sendRedirect("index.jsp");                         //重定向页面到 index.jsp
%>
```

秘笈心法

虽然客户端长时间不向服务器发送请求后，session 对象会自动消失，但对于某些实时统计在线人数的网站（如聊天室），每次都等 session 过期后，才能统计出准确的人数，这是远远不够的。所以还需要手动销毁 session。通过 session 对象的 invalidate()方法可以销毁 session，语法格式如下：

```
session.invalidate();
```

session 对象被销毁后，将不可以再使用该 session 对象。如果在 session 被销毁后，再调用 session 对象的任何方法，都将报出 Session already invalidated 异常。

实例 114　统计用户在站点停留的时间

光盘位置：光盘\MR\04\114

初级　实用指数：★★

实例说明

用户在站点的停留时间可以通过计算从 session 对象的创建到 session 对象的销毁之间的时间差来得到。运行本实例，系统会在用户下线时把用户的登录、离开和停留时间写进数据表中，如图 4.24 所示。

user_ip	user_ontime	user_offtime	user_stoptime
192.168.1.38	2006-10-13 10:43:37	2006-10-13 10:44:41	0小时1分4秒
192.168.1.12	2006-10-13 10:45:22	2006-10-13 10:45:43	0小时0分21秒
192.168.1.16	2006-10-13 10:45:53	2006-10-13 10:46:46	0小时0分52秒
192.168.1.22	2006-10-13 10:46:49	2006-10-13 10:47:48	0小时0分59秒
192.168.1.14	2006-10-13 12:51:13	2006-10-13 12:51:39	0小时0分26秒
192.168.1.12	2006-10-20 11:46:48	2006-10-20 11:47:21	0小时0分33秒
192.168.1.47	2006-10-27 05:12:12	2006-10-27 05:12:31	0小时0分18秒
127.0.0.1	2006-10-27 05:14:54	2006-10-27 05:15:38	0小时0分43秒

图 4.24　统计用户在站点停留的时间

关键技术

本实例主要继承 HttpSessionBindingListener 接口监听会话的创建与销毁。继承该接口的类必须实现继承的抽象方法 valueBound(HttpSessionBindingEvent)和 valueUnbound(HttpSessionBindingEvent)。前者在向 session 中存入实现该接口的类对象时触发，后者在从 session 移除该对象时触发。

设计过程

（1）创建对数据库进行操作的 DB 类文件，关键代码如下：

```
public class DB {
 private Connection con;
 private Statement stm;
 private ResultSet rs;
 private String classname="com.microsoft.jdbc.sqlserver.SQLServerDriver";
 private String url="jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=db_database08";
 public DB(){}
 public Connection getCon(){
        try{
               Class.forName(classname);                    //加载数据库驱动类
               con=DriverManager.getConnection(url,"sa",""); //获取数据库连接
        }catch(Exception e){
               e.printStackTrace();
        }
        return con;
 }
 public Statement getStm(){
        try{
               con=getCon();                                //获取数据库连接
        stm=con.createStatement();                          //获取 stm 对象
        }catch(Exception e){e.printStackTrace(System.err);}
        return stm;                                         //返回 stm 对象
 }
 public int dosql(String sql){                              //执行 SQL 语句
        int i=-1;
        getStm();                                           //获取 stm 对象
        try{
               i=stm.executeUpdate(sql);                    //执行 SQL 语句，获取执行结果
        }catch(Exception e){e.printStackTrace();}
        return i;                                           //返回执行结果
 }
}
```

（2）创建 StopTime 类，该类用来计算停留的时间，关键代码如下：

```
package com.count.stoptime;
import java.util.*;
import java.text.*;
public class StopTime {
     private int h=0;
     private int m=0;
     private int s=0;
     private Date start=null;                               //创建时间
     private Date end=null;                                 //结束时间
     private SimpleDateFormat format=new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
     public StopTime(){}
     public void setStart(Date start){
          this.start=start;
     }
     public void counttime(Date end){                       //计算停留时间的方法
          his.end=end;
          long howmuch=this.end.getTime()-start.getTime();  //计算出相差的毫秒数
          h=(int)(howmuch/1000/60/60);                      //计算出小时
          howmuch=howmuch-h*60*60*1000;                     //剩余的毫秒数
          m=(int)(howmuch/1000/60);                         //计算出分钟
          howmuch=howmuch-m*60*1000;
          s=(int)(howmuch/1000);                            //计算出秒数
     }
```

```
    public String getTimes(){
        String times=this.h+"小时"+this.m+"分"+this.s+"秒";
        return times;
    }
    public String getStart(){
        return format.format(start);
    }
    public String getEnd(){
        return format.format(end);
    }
}
```

（3）创建 UserLine 类，该类实现 HTTP 会话的监听，关键代码如下：

```
package com.count.stoptime;
import java.util.*;
import javax.servlet.http.HttpSessionBindingEvent;
public class UserLine implements javax.servlet.http.HttpSessionBindingListener{
    private String userip;
    private StopTime stoptime=new StopTime();
    public UserLine(){
        username="";
    }
    …//省略了属性的 setXXX()和 getXXX()方法
     public void valueBound(HttpSessionBindingEvent arg0){
         stoptime.setStart(new Date());              //记录创建的时间
         System.out.println(this.userip+"于"+new Date().toLocaleString()+"上线！ ");
    }
    public void valueUnbound(HttpSessionBindingEvent arg0){
        stoptime.counttime(new Date());             //计算停留时间
        System.out.println(this.userip+"于"+new Date().toLocaleString()+"下线！ ");
        writetime();                                //将信息写入数据库中
    }
    public void writetime(){
        String starttime=stoptime.getStart();
        String endtime=stoptime.getEnd();
        String times=stoptime.getTimes();
        String sql="insert into tb_stoptime values('"+starttime+"','"
                                            +endtime+"','"+times+"')";
        DB db=new DB();
        db.dosql(sql);
        db.closed();
    }
}
```

（4）创建站点的首页面 index.jsp。当用户访问该页面时，先设置 session 的有效活动时间，当将实现会话监听的类对象存入 session 对象后，触发监听器的 valueBound()方法记录用户的登录时间，关键代码如下：

```
<%@ page import="com.count.stoptime.*" %>
<%
  session.setMaxInactiveInterval(10);          //设置 session 的有效活动时间
  String userip=request.getRemoteAddr();       //获得用户的 IP 地址
  UserLine userline=new UserLine();            //实例化一个监听类对象
  userline.setUserip(userip);                  //设置 IP
  session.setAttribute("logonuser",userline);  //将实例化的监听类对象存入 session 对象中
%>
    <table>
      <tr bgcolor="lightgrey" height="25">
        <td align="center">
          欢迎登录！
        </td>
      </tr>
      <tr>
        <td>
          本实例用于统计用户在站点的停留时间。<br>
          该时间是一个 session 对象从创建到结束期间的时间差。<br>
          本实例应用了 Servlet 事件监听中对会话监听的方法。<br>
          主要是通过继承 HttpSessionBindingListener 接口监听对 HTTP 会话操作！
         </td>
        </tr>
     </table>
```

秘笈心法

HttpSessionBindingListener 接口能够监听会话与一个属性绑定或解除绑定的事件，当 Web 应用把一个属性与会话绑定后，Servlet 容器会调用该接口的 valueBound()方法，当 Web 应用将要把一个属性与会话解除绑定之前，Servlet 容器会调用 valueUnbound()方法。

实例 115 判断用户是否在线

光盘位置：光盘\MR\04\115

初级

实用指数：★★

实例说明

本实例中判断用户是否在线是通过对 HTTP 会话的监听来实现的。运行本实例，用户可以输入任意用户名进行登录，如果用户名正在使用，则提示“该用户已在线”，否则进入如图 4.25 所示的页面。

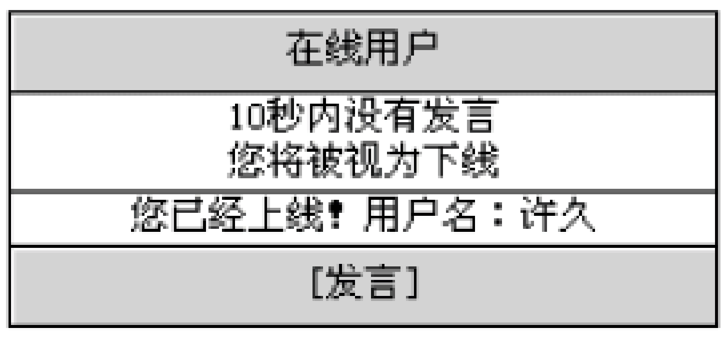


图 4.25 判断用户是否在线

关键技术

通过继承 HttpSessionBindingListener 接口实现对 HTTP 会话的监听。继承该接口的类必须实现继承的抽象方法 valueBound(HttpSessionBindingEvent)和 valueUnbound(HttpSessionBindingEvent)。前者在向 session 中存入实现该接口的对象时触发，后者在从 session 移除该对象时触发。

设计过程

（1）创建监听 HTTP 会话的 OnLine 类文件，具体代码如下：

```
package com.line;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.HttpSession;
import javax.servlet.http.HttpSessionBindingEvent;
import javax.servlet.http.HttpSessionBindingListener;
public class OnLine implements HttpSessionBindingListener{
    private String username;                          //用户输入的用户名
    private boolean mark=true;
    private Vector vc=new Vector();
    public void setUsername(String username){
        this.username=username;
    }
    public void valueBound(HttpSessionBindingEvent se) {
        HttpSession session=se.getSession();
        ServletContext sct=session.getServletContext();   //该对象可被所有页面共享
        vc=(Vector)sct.getAttribute("online");
        if(vc==null){                                     //服务器启动后，第一个用户进行访问
            vc=new Vector();
            vc.add(username);
        }else{
            if(vc.contains(username)){                    //如果存在该用户
                mark=false;
            }else
                vc.add(username);
        }
        sct.setAttribute("online",vc);
    }
    public void valueUnbound(HttpSessionBindingEvent se) {
        HttpSession session=se.getSession();
        ServletContext sct=session.getServletContext();
        vc=(Vector)sct.getAttribute("online");
        if(vc!=null){
```

```
                vc.removeElement(this.username);        //用户下线时移除该用户
            }
        }
    public boolean getMark(){
        return this.mark;
    }
}
```

（2）创建 UserLogOn 类文件，其中的 checkuser()方法用来验证用户信息并返回一个 boolean 类型值。该类在用户名不为空的情况下，调用 OnLine 类中的方法进行相应的操作，关键代码如下：

```
package com.line;
import javax.servlet.http.HttpSession;
public class UserLogOn {
    private String username;
    private String backstr="";
    private OnLine on=new OnLine();
    public UserLogOn(){}
    …//省略了 username 属性的 setXXX()和 getXXX()方法
    public boolean checkuser(HttpSession session){
        boolean mark=true;
        if(this.username==null||this.username.equals("")){
            this.backstr="<li>请输入<b>用户名！</b></li><br>";
            mark=false;
        }
        if(mark){
            on.setUsername(this.username);
            session.setAttribute("onlineuser",on);    //将 OnLine 监听类对象存入 session 中
            mark=on.getMark();
            if(!mark)
                this.backstr="该用户已在线！";
        }
        return mark;
    }
    public String getBackstr(){
        return this.backstr;
    }
}
```

（3）创建首页面 index.jsp 供用户输入登录信息，关键代码如下：

```
<form action="dologon.jsp">
  <table>
    <tr bgcolor="lightgrey" height="30">
      <td align="center">用户登录</td>
    </tr>
    <tr height="30">
      <td align="center">
        用户名：<input type="text" name="username" size="30">
      </td>
    </tr>
    <tr bgcolor="lightgrey">
      <td align="center">
        <input type="submit" name="logon" value="登录">
        <input type="reset" name="clear" value="重置">
      </td>
    </tr>
  </table>
</form>
```

（4）创建接收 Form 表单的页面 dologon.jsp。在该页面中调用 UserLogOn 类中的 checkuser()方法来验证用户身份。如果输入的用户名不为空并且该用户不在线，则转向 myline.jsp 页面，关键代码如下：

```
<jsp:useBean id="mylogon" class="com.line.UserLogOn"/>
<%
  String username=request.getParameter("username");
  if(username==null)
      username="";
      username=new String(username.getBytes("ISO-8859-1"),"gbk");
```

```
  mylogon.setUsername(username);           //设置用户名
  if(mylogon.checkuser(session)){           //如果用户名不为空并且该用户不在线
     session.setAttribute("username",username);
     response.sendRedirect("myline.jsp");
  }
%>
     <table>
       <tr bgcolor="lightgrey" height="30">
         <td align="center">登录状况</td>
       </tr>
       <tr height="50">
         <td align="center">
           <%=mylogon.getBackstr()%>
         </td>
       </tr>
     </table>
```

（5）创建 myline.jsp 页面，该页面用于显示用户的在线情况，关键代码如下：

```
<%@ page import="java.util.*"%>
<%
  session.setMaxInactiveInterval(10);
  Vector vec=(Vector)application.getAttribute("online");
%>
    ……
    <meta http-equiv="refresh" content="12">
    ……
  <table>
    <tr>
      <td width="60%" align="center" valign="middle">10 秒内没有发言<br>您将被视为下线</td>
    </tr>
    <tr>
      <td align="center" valign="middle">
      <%
      if(vec==null||vec.size()==0)
         out.println("没有用户！");
      else{
         if(vec.contains(session.getAttribute("username"))){
            out.println("您已经上线！用户名："+session.getAttribute("username"));
         }
         else
            out.println("您没有上线！");
         }
      %>
      </td>
    </tr>
    <tr bgcolor="lightgrey">
      <td align="center" height="25" colspan="2">
      <%
      if(session.getAttribute("username")==null)     //表示 session 过期已销毁
         out.println("您已经下线！");
      else
         if(vec.contains(session.getAttribute("username")))
            out.println("<a href='myline.jsp'>[发言]</a>");
         else
         out.println("请先登录！");
      %>
  </table>
  <%
    if(!vec.contains(session.getAttribute("username")))
       out.println("<a href='index.jsp'>[登录]</a>");
  %>
```

秘笈心法

本实例另外设置一个 Vector 对象，并将它存入 Application 对象中，以用来存储在线的用户。当用户登录时，如果用户名存在于 Vector 对象中，则提示“该用户已在线”信息，否则将该用户名存入 Vector 对象中；当用户下线时，从 Vector 对象中移除该用户名。

实例 116　实时统计在线人数

光盘位置：光盘\MR\04\116

初级

实用指数：★★

实例说明

本实例主要实现显示正在访问网站人数的功能，它可以使管理员知道网站当前被访问的情况，以此来了解网站的受关注程度。运行本实例，页面中将显示正在访问网站的所有用户，并统计出在线人数，实例运行效果如图 4.26 所示。

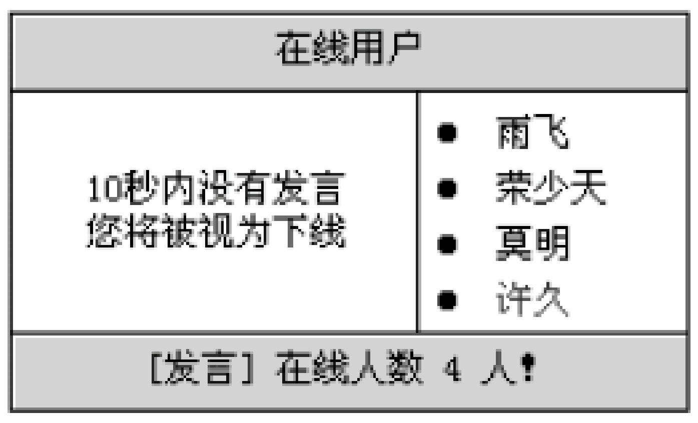


图 4.26　实时统计在线人数

关键技术

本实例应用的方法介绍如下。

（1）Vector 对象的 elements()方法

语法格式如下：

```
Enumeration em=VectorObject.elements()
```

功能：返回 Vector 对象中的所有值，其结果是一个 Enumeration 枚举对象。

参数说明

❶ em：表示一个 Enumeration 枚举对象。

❷ VectorObject：表示实例化的 Vector 对象。

（2）EnumerationObject 对象的 nextElement()方法

语法格式如下：

```
EnumerationObject.nextElement()
```

功能：遍历枚举对象中的值。使用该方法获得的值应强制转换为该值原来的类型。

（3）Enumeration 枚举对象的 hasMoreElements()方法

语法格式如下：

```
EnumerationObject.hasMoreElements()
```

功能：判断枚举对象中是否还存在值，该方法返回一个 boolean 类型值。

其中参数 EnumerationObject 为实例化的枚举对象。

设计过程

（1）创建用于监听 HTTP 会话的 OnLine 类文件和用于验证登录信息的 LogOnNum 类文件。由于这两个类的实现技术与实例 115 中的 OnLine 类和 User LogOn 类相同，在此不再赘述。

（2）创建一个首页面 index.jsp 供用户输入登录信息。相关代码可参见实例 115。

（3）创建接收 Form 表单的页面 dologon.jsp。相关代码可参见实例 115 中的 dologon.jsp 文件或光盘。

（4）创建 online.jsp 页面，该页面用于显示所有在线用户的信息，关键代码如下：

```
<%@ page import="java.util.*"%>
<%
  session.setMaxInactiveInterval(10);
  Vector vec=(Vector)application.getAttribute("online");
%>
     <table>
      <tr>
        <td align="left" valign="middle">
          <table>
          <%
            if(vec==null||vec.size()==0)
               out.println("<tr><td align='center'>没有用户！ </td></tr>");
            else{          //显示所有在线用户
               Enumeration em=vec.elements();
               while(em.hasMoreElements()){
                    String username=(String)em.nextElement();
```

```
                    if(username.equals(session.getAttribute("username")))
                        out.println("<tr><td align='left'><li><font color='red'>"+username+"</font></td></tr>");
                    else
                        out.println("<tr><td align='left'><li>"+username+"</td></tr>");
                }//while
            }
            %>
            </table>
        </td>
    </tr>
    <tr bgcolor="lightgrey">
        <td align="center" height="25" colspan="2">
          <%
            if(session.getAttribute("username")==null){
                out.println("您已经下线！");
            }
            else{
                if(vec.contains(session.getAttribute("username"))){
                    out.println("<a href='online.jsp'>[发言]</a>");
                    out.println("在线人数 "+vec.size()+" 人！");
                }
                else
                    out.println("请先登录！");
                }
        %>
</table>
<%
if(!vec.contains(session.getAttribute("username")))
    out.println("<a href='index.jsp'>[登录]</a>");
%>
```

秘笈心法

本实例主要是通过 HttpSessionBindingListener 接口来监听 session 会话的绑定状态，当用户登录时，会将用户名保存到 session 中，这时就会触发 HttpSessionBindingListener 接口的 valueBound()方法，然后将用户名保存在 Vector 中，再将 Vector 存储在 ServletContext 中，最后在页面中读取出用户名信息并显示，此时设置 session 的有效期为 10 秒，当 session 超时后，就会触发 HttpSessionBindingListener 接口的 valueUnBound()方法，然后从 Vector 中移出当前的用户。

4.3 JSP 的自定义标签

JSP 中设计自定义标签的目的就是为了实现 HTML 代码的重用，许多公司已经有属于自己的自定义标签，将多样化的表格、复杂的报表等封装成自定义标签，并逐步完善形成一个平台，在此基础上继续进行程序开发，大大提高了开发的效率以及风险控制能力。

实例 117	带标签体的自定义标签 光盘位置：光盘\MR\04\117	初级 实用指数：★★

实例说明

本实例主要使用标签体的 getBodyContent()方法自定义一个标签，把一段字母按大写输出，运行结果如图 4.27 所示。

关键技术

使用 BodyTagSupport 类，再调用 PageContent()方法，BodyTag 接口除了常用的 doStartTag()、doEndTag()之外，还有 doInitBody()，该方法每次处理完标签体都要被执行，重载这个方法来处理自己的标签体，这个方法通

常返回 SKIP_BODY_TAG，指明没有标签体需要继续处理，如果返回的是 EVAL_BODY_TAG，则这个标签体被再次处理，导致调用 doAfterBody()，直到 doAfterBody 返回 SKIP_BODY。



图 4.27　按大写输出结果

设计过程

（1）创建 BodyTagSupport 类的实现类 TagBody.java，主要代码如下：

```
public class TagBody extends BodyTagSupport {
 public int doEndTag() throws JspException {
        String ct = this.getBodyContent().getString();
        //获取标签体的代码
        try {
                //以大写输出
                this.pageContext.getOut().print(ct.toUpperCase());
        } catch (IOException e) {
        }
        return EVAL_PAGE;
 }
}
```

（2）创建 tagbody.tld 文件，主要代码如下：

```
<tlib-version>1.0</tlib-version>
 <short-name>toUpperCase</short-name>
        <tag>
                <name>toupp</name>
                <tagclass>com.mr.bodycontent.TagBody</tagclass>
                <bodycontent>JSP</bodycontent>
 </tag>
```

（3）创建 index.jsp 页面，主要代码如下：

```
<body>
  <toUpperCase:toupp>hello welcome to mr networkschool!!!!</toUpperCase:toupp>
 </body>
```

秘笈心法

当执行 doStartTag()方法时，返回值若是 EVAL_BODY_BUFFERED，则不会输出；返回值如果是 EVAL_BODY_INCLUDE，则会输出标签体内容。标签体的内容是通过 setBodyContent()方法添加到标签类中的，可以使用 getBodyContent()方法得到标签体的内容。

实例 118	自定义多次执行的循环标签 光盘位置：光盘\MR\04\118	初级 实用指数：★★

实例说明

本实例定义了一个 numbers 变量进行自减运算，使用 doAfterBody()方法来循环然后在页面中输出苹果，效

果如图 4.28 所示。

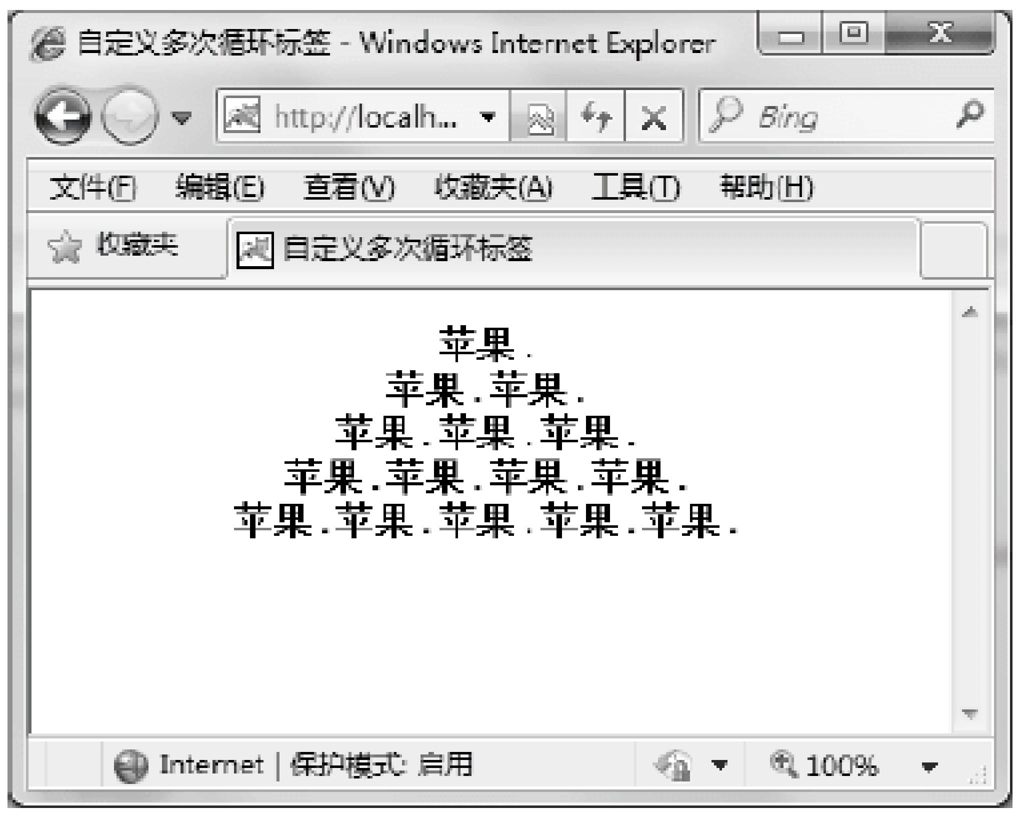


图 4.28　运行结果

关键技术

本实例主要使用 BodyTagSupport 类的 doAfterBody()方法，如果该方法返回 Tag.SKIP_BODY，就不再执行标签主体的内容；如果该方法返回 IterationTag.EVAL_BODY_AGAIN，就继续重复执行标签主体的内容。

设计过程

（1）创建实现类 Mul 类，用 doAfterBody()方法来控制单次执行还是多次执行，主要代码如下：

```
public class Mul extends BodyTagSupport {
 private int numbers;                        //定义变量 numbers
 public int doStartTag() throws JspException {
        numbers = 5;                         //初始值为 5
        return super.doStartTag();
 }
 public int doAfterBody() throws JspException {
        if (numbers-- > 0) {                 //自减运算
               try {
                      this.getPreviousOut()
                                  .println(this.getBodyContent().getString());
               } catch (Exception e) {
               }
               return EVAL_BODY_AGAIN;
        } else {
               return SKIP_BODY;
        }
 }
}
```

（2）创建 mul.tld 标签，代码如下：

```
<tlib-version>1.0</tlib-version>
 <short-name>tagmul</short-name>
<jspversion>1.1</jspversion>
   <tag>
               <name>muls</name>
               <tag-class>com.zm.Mul</tag-class>
               <body-content>JSP</body-content>
</tag>
```

（3）创建 index.jsp 页面，使用<pre>迭代，主要代码如下：

```
<h3 align="center">
<pre><tagmul:muls>苹果</tagmul:muls></pre>
</h3>
```

秘笈心法

带标签体的标签大体可以分为单次执行（single Evaluation）和多次执行（multiple Evaluation）。单次执行指的是标签体只会被使用一次，而多次执行是指标签体可以被使用多次。

实例 119　自定义显示版权信息标签

光盘位置：光盘\MR\04\119

初级

实用指数：★★

实例说明

本实例通过自定义一个标签为 Copyright 和相对应的 java 类来实现，此类必须实现 Tag 接口，在 JSP 中执行自定义的标签<taglib:copyright />来输出定义好的版权内容，运行结果如图 4.29 所示。

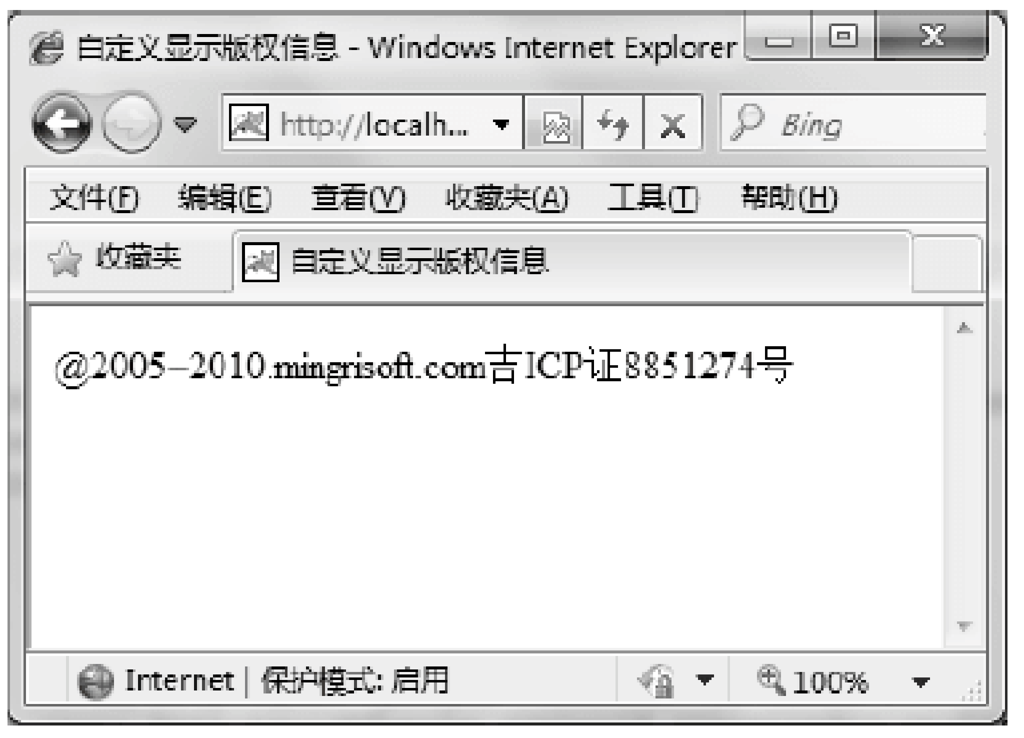


图 4.29　版权信息

关键技术

所有的标签处理类都要实现 JspTag 接口。这个接口只是一个标识接口，没有任何方法，主要是作为 Tag 和 SimpleTag 接口的共同接口。在 JSP 2.0 以前，所有的标签处理类都要实现 Tag 接口，实现该接口的标签为传统标签。Tag 接口定义了所有传统标签处理类都要实现的基本方法，包括以下几种。

- setPageContext(PageContext pc)：由 Servlet 容器调用该方法，向当前标签处理对象（即 Tag 对象）传递当前的 PageContext 对象。
- setParent(Tag t)：由 Servlet 容器调用该方法，向当前 Tag 对象传递父标签的 Tag 对象。
- getParent()：返回 Tag 类型的父标签的 Tag 对象。
- release()：当 Servlet 容器需要释放 Tag 对象占用的资源时，会调用此方法。
- doStartTag()：当 Servlet 容器遇到标签的起始标志时，会调用此方法。
- doEndTag()：当 Servlet 容器遇到标签的结束标志时，会调用此方法。

设计过程

（1）创建一个 Copyright 类，主要代码如下：

```
public class copyright implements Tag {
private PageContext pageContext;                    //在 JSP 中显示的内容
public PageContext getPageContext() {
        return pageContext;
}
public void setPageContext(PageContext pageContext) {
        this.pageContext = pageContext;
}
public int doEndTag() throws JspException {         //标签结束时执行
        JspWriter out = pageContext.getOut();       //获取 out 对象
        try {
                //输出已定义的版权信息
                out.println(ResourceBundle.getBundle("copyright").getString("copyright"));
        } catch (IOException e) {
                throw new JspException(e);
        }
        return EVAL_PAGE;
}
public int doStartTag() throws JspException {       //标签开始时执行
```

```
        return SKIP_BODY;
}
public void release() {
}
public Tag getParent() {
        return null;
}
public void setParent(Tag arg0) {
}
}
```

（2）创建 tld 标签库描述文件，在/WEB-INF/下新建 taglib.tld 并添加以下代码：

```
<?xml version="1.0" encoding="UTF-8"?>
<taglib version="2.0" xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee web-jsptaglibrary_2_0.xsd">
<tlib-version>1.0</tlib-version>
<jspversion>1.1</jspversion>
<short-name>taglib</short-name>
<uri>http://mingrisoft.com/tags</uri>
<tag>
        <name>copyright</name>
        <tagclass>com.mr.tags.copyright</tagclass>
        <bodycontent>JSP</bodycontent>
</tag>
</taglib>
```

（3）创建 copyright.jsp 页面，并添加刚定义好的标签，主要代码如下：

```
<body>
        <taglib:copyright></taglib:copyright>
 </body>
```

秘笈心法

标签处理类的对象（Tag 对象）由 Servlet 容器负责创建。Servlet 容器在执行 JSP 文件时，如果遇到 JSP 文件中的自定义标签，就会寻找缓存中的相关 Tag 对象；如果还不存在，就创建一个 Tag 对象，把它存放在缓存中，以便下次处理自定义标签时重复使用。

实例 120　自定义图片浏览标签

光盘位置：光盘\MR\04\120

初级　实用指数：★★

实例说明

本实例介绍如何从服务器上获得图片文件并显示给客户端。运行本实例，在页面中将显示服务器中的指定图片，如图 4.30 所示。

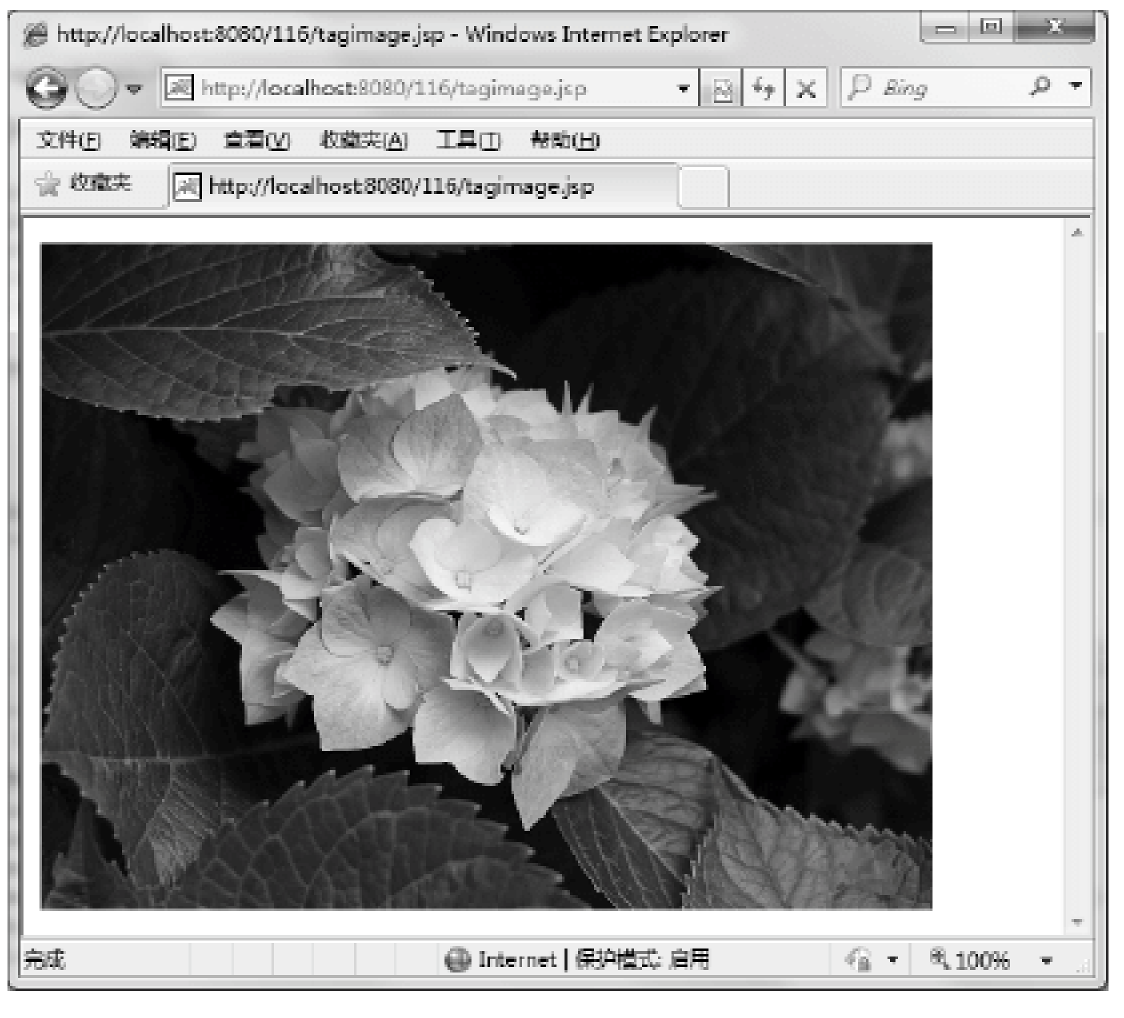


图 4.30　图片浏览标签

关键技术

通过 BufferedInputStream 建立输入流完成读取文件的操作，其语法格式如下：

```
FileInputStream in = new FileInputStream(new File(fileName))
```

参数说明

❶ fileName：文件的名称。

❷ new File(fileName)：建立文件名称为 fileName 的文件。通过类 BufferedInputStream 建立缓冲输入流，它的参数为 FileInputStream。

设计过程

（1）创建 imagetag.java 类文件，此类继承自 SimpleTagSupport 类。在该类中分别定义了两个变量 fileName 和 response，表示图片的名称和给予客户端的响应，并在 doTag()方法中完成图片数据的传输，关键代码如下：

```
public class imagetag extends SimpleTagSupport {
	private String fileName;
	private HttpServletResponse response;
	public void setFileName(String fileName) {
		this.fileName = fileName;
	}
	public void setResponse(HttpServletResponse response) {
		this.response = response;
	}
	public void doTag(){
		try {
			FileInputStream in = new FileInputStream(new File(fileName));	//创建输入流完成读取文件
			BufferedInputStream bufferin = new BufferedInputStream(in);	//创建缓冲输入流参数为 in
			OutputStream out   = response.getOutputStream();	//向客户端输出流
			BufferedOutputStream bufferout = new BufferedOutputStream(out);	//创建缓冲输出流参数为 out
			byte b[] = new byte[1024*5];
			for(int i =bufferin.read(b);i!=-1;){
				bufferout.write(b);
				bufferin.read(b);
			}
			bufferout.flush();
			bufferout.close();
			bufferin.close();
			out.close();
			in.close();
		} catch (FileNotFoundException e) {
			e.printStackTrace();
		} catch (IOException e) {
			e.printStackTrace();
		}
	}
}
```

（2）创建标签描述文件 example.tld，相应地描述标签类的配置，关键代码如下：

```
<tlib-version>1.0</tlib-version>
<short-name>filetag</short-name>
<tag>
		<name>filetag</name>
		<tag-class>com.mr.imagetag</tag-class>
		<body-content>empty</body-content>
	<attribute>
		<name>fileName</name>
		<required>true</required>
		<rtexprvalue>true</rtexprvalue>
	</attribute>
	<attribute>
		<name>response</name>
		<required>true</required>
		<rtexprvalue>true</rtexprvalue>
		</attribute>
</tag>
```

（3）在 web.xml 中配置引导标签类的地址和 tld 文件的位置，关键代码如下：

```
<jsp-config>
        <taglib>
                <taglib-uri>/http://www.mingrisoft.com/tags</taglib-uri>
                <taglib-location>/WEB-INF/example.tld</taglib-location>
        </taglib>
   </jsp-config>
```

（4）创建 tagimage.jsp 页面，它的作用是引入自定义的图片标签，然后读取服务器上的图片并将其显示在页面上，关键代码如下：

```
<body>
        <filetag:filetag response="${pageContext.response}"
                fileName='<%=pageContext.getServletContext().getRealPath("Hd.jpg") %>' />
 </body>
```

秘笈心法

文件流读取完毕后，应该调用其 close()方法及时关闭流，以免造成不必要的异常和占用不必要的内存空间。

实例 121 自定义文件下载的标签

光盘位置：光盘\MR\04\121

初级

实用指数：★★

实例说明

进行 Web 开发或者网站设计时，数据下载是经常会用到的功能，本实例通过 SimpleTag 类设计一个通用的文件下载标签，完成文件下载功能。运行本实例，如图 4.31 所示，单击“下载”超链接即可下载指定文件。

编号	文件名称	文件大小	文件类型	文件路径	文件下载
01	JavaScript快速查询手册	3.52 MB	pdf	\download\	下载
02	Spring开发参考手册	1.38 MB	pdf	\download\	下载
03	程序员超级开发宝典	4.22 MB	chm	\download\	下载
04	MySQL 4.1.0中文参考手册	1.39 MB	chm	\download\	下载
05	J2EE OA项目开发日记	845 KB	doc	\download\	下载
06	J2EE体系结构设计	426 KB	doc	\download\	下载
07	apache-tomcat-5.5.16	5.88 MB	exe	\download\	下载
08	apache-tomcat-5.5.16-admin	2.23 MB	zip	\download\	下载
09	AdbeRdr60中文正式版	24.3 MB	exe	\download\	下载

图 4.31　自定义文件下载标签

关键技术

在制作文件下载标签时，最关键的步骤是实现文件下载功能，本实例实现文件下载的基本思路如下：

（1）通过 request 类中的 getRealPath()方法获得文件的真实路径，即文件在服务器上的绝对路径。

（2）通过 response 类中的 setHeader()方法设置客户端文件下载的路径。

（3）通过文件类 BufferedInputStream 进行数据的读入。

（4）通过 BufferedOutputStream.java 类进行数据的写入，实现文件的下载。

设计过程

（1）创建 DownTag.java 类文件，此类继承自 SimpleTagSupport 类，关键代码如下：

```
package com;
import java.io.*;
import javax.servlet.*;
public class DownTag extends SimpleTagSupport {
```

首先，创建存放文件的名称及相应的 HttpServletResponse 和 HttpServletRequest 类对象的属性，并且设置相对应的 get()/set()方法，关键代码如下：

```
private String fileName;                                    //文件的名称
private String fileType;                                    //文件的类型
private HttpServletResponse response;                       //HttpServletResponse 的属性名称
private HttpServletRequest request;                         //HttpServletRequest 的属性名称
private String filePath=null;                               //文件的路径
public void setRequest(HttpServletRequest request) {
this.request = request;
}
public void setFileName(String filePath) {
this.fileName = filePath;
}
public void setFileType(String fileType) {
this.fileType = fileType;
}
public void setResponse(HttpServletResponse response) {
this.response = response;
}
```

然后，在 doTag()方法中编写实现文件下载的代码，先设置客户端要下载文件的位置，接着定义输入缓冲流和输出缓冲流进行文件的读写操作，关键代码如下：

```
public void doTag() throws JspException{
 filePath = request.getRealPath("/");
 response.setHeader("Content-disposition", "attachment; filename=" + fileName + "." + fileType);
 try {
        fileName = new String(fileName.getBytes("ISO-8859-1"),"GBK");              //文件名转换编码
        String convertfilepath = filePath.trim() + "download\\"   + fileName.trim() + "." + fileType.trim();
        File file = new File(convertfilepath.replaceAll("\\\\", "/"));            //替换路径中的斜杠
        FileInputStream input = new FileInputStream(file);                        //实例化文件输入流
        BufferedInputStream bufferinput = new BufferedInputStream(input);          //带缓冲文件输入流
        ServletOutputStream out = response.getOutputStream();                     //获取文件输出流
        BufferedOutputStream bufferoutput = new BufferedOutputStream(response.getOutputStream());   //封装文件输入流
        byte[] temp = new byte[2048];
        int bytesRead;
        while ((bytesRead = (input.read(temp, 0, temp.length))) != -1 ) {
              bufferoutput.write(temp, 0, bytesRead);
        }
        bufferoutput.flush();                                                      //刷新
        if (input != null) {
              input.close();                                                       //关闭输入流
        }
        if (bufferoutput != null) {
              out.close();                                                         //关闭文件输出流
              bufferoutput.close();                                                //关闭缓冲输出流
        }
 } catch (Exception e) {
        e.printStackTrace();
 }
}
```

（2）在 WEB-INF 目录下编写 down.tld 文件，完成标签的配置，在这个配置中设置标签中所使用的类文件以及这个类中的私有变量，关键代码如下：

```
<tag>
   <name>tagfile</name>
<tag-class>com.DownTag</tag-class>
<body-content>empty</body-content>
<attribute>
   <name>fileName</name>
   <required>true</required>
   <rtexprvalue>true</rtexprvalue>
</attribute>
<attribute>
<name>fileType</name>
   <required>true</required>
   <rtexprvalue>true</rtexprvalue>
</attribute>
<attribute>
   <name>response</name>
   <required>true</required>
```

```
    <rtexprvalue>true</rtexprvalue>
</attribute>
<attribute>
    <name>request</name>
    <required>true</required>
    <rtexprvalue>true</rtexprvalue>
</attribute>
</tag>
```

（3）在 web.xml 文件中添加所使用的标签，关键代码如下：

```
<taglib>
<taglib-uri>/downtag</taglib-uri>
<taglib-location>/WEB-INF/down.tld</taglib-location>
</taglib>
```

（4）在 Web 根目录下建立一个 index.jsp 文件，在这个文件中连接数据库并显示数据库中要下载的文件信息，关键代码如下：

```
<form action="" method="post">
    <table width="639" border="1" cellspacing="0" cellpadding="0">
      <tr>
          <td width="54">编号</td>
        <td width="244">文件名称</td>
        <td width="94">文件大小</td>
<td width="64">文件类型</td>
<td width="92">文件路径</td>
<td width="77">文件下载/td>
       </tr>
       <c:forEach var="collection" items="${collection}">
       <tr>
       <td><c:out value="${collection[0]}"/></td>
<td><c:out value="${collection[1]}"/></td>
<td><c:out value="${collection[2]}"/></td>
<td><c:out value="${collection[3]}"/></td>
<td><c:out value="${collection[4]}"/></td>
<td align="center"><a href="downfile.jsp?name=${collection[1]}&type=${collection[3]}">下载</a></td>
    <down:tagfile />
       </tr>
       </c:forEach>
    </table>
  </form>
```

（5）在 Web 根目录下创建 downfile.jsp 文件，完成文件下载功能，关键代码如下：

```
<down:tagfile fileName="${param['name']}" fileType="${param['type']}" response="<%=response%>" request="<%=request%>"/>
```

秘笈心法

在实现文件的下载功能时，应用该自定义标签即可，这样不仅可以提高代码的可重用性，而且能够提高项目的开发效率，避免在 JSP 中编写大量的 Java 代码。

实例 122	自定义数据查询的标签 光盘位置：光盘\MR\04\122	初级 实用指数：★★

实例说明

本实例将介绍如何通过 SimpleTag 来制作通用的标签。该方法实现的标签可以移植到任何一个 JSP 页面中，实例中定义的标签接收 SQL 查询语句作为参数，在使用时只需更改相应的参数即可实现不同的数据查询功能，实例运行结果如图 4.32 所示，页面中显示了标签返回的用户查询结果。

关键技术

通过 JspWriter.java 类向页面输出表格，然后在表格中进行数据填充。获取 JspWriter 对象的语法格式如下：

```
JspWriter out = this.getJspContext().getOut();
```

姓名	性别	年龄	身份证号	政治面貌	家庭电话	家庭地址
张三	男	21	2201048402154	团员	1594306****	长春市亚太大街
李四	男	22	2201048008171	团员	1597*****	长春市解放大路
王五	男	23	2201048502155	团员	13578******	长春市青年路
赵六	女	18	2201048705134	党员	87965***	长春市新民大街

图 4.32　自定义数据查询标签运行结果

设计过程

（1）创建 DBCon.java 类，用于创建数据库连接，关键代码如下：

```
public class DBCon {
 public static Connection getConnection(){
         Connection conection = null;
         try{

                 Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver");          //加载数据库驱动类
                 conection = DriverManager.getConnection("jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=db_database17 ","sa","");//创建连接
       }catch(Exception e){
         e.printStackTrace();
       }
       return conection;
 }
}
```

（2）创建 TableTag.java 类文件，此类继承自 SimpleTag 类，关键代码如下：

```
import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.naming.*;
public class TableTag extends SimpleTagSupport {
//定义两个变量分别用来存储表格的标题名称和查询的 SQL 语句
private String tableTitle[];                                                    //标题名称定义为数组
public void setTableTitle(String[] title) {
       this.tableTitle = title;
}
private String sqlSelect;                                                       //定义 SQL 语句
public void setSqlSelect(String sql) {
       this.sqlSelect = sql;
}
public void doTag() {
//获取数据库的连接并执行 SQL 语句
connection = DBCon.getConnection();                                             //获取数据库连接
PreparedStatement pstmt = connection.prepareStatement(this.sqlSelect);          //执行 SQL 语句
java.sql.ResultSet resultset = pstmt.executeQuery();                            //获取结果集
ResultSetMetaData rsmd = resultset.getMetaData();                               //获取表结构对象
//显示表格的标题
          out.write("<table border=1 align='center'><tr bgcolor='#669966'   align='center'>");
          for (int i = 0 ; i < tableTitle.length ; i++){
                 try {
                          out.write("<td height='28'>" + tableTitle[i] + "</td>");       //输出表头
                 } catch (IOException e) {
                          e.printStackTrace();
                 }
          }
          out.write("</tr>");
          while(resultset.next()){                                              //循环输出表中的记录
                out.write("<tr align='center' height='28px' >");
                for ( int i = 1 ; i <= rsmd.getColumnCount() ; i ++){
                 out.write("<td>" + resultset.getObject(i).toString() + "</td>");
                }
                out.write("</tr>");
          }
          out.write("</table>");
```

```
    }catch(Exception sql){
        sql.printStackTrace();
    } finally{
        try {
            if(connection!=null)
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

（3）在 WEB-INF 目录下建立 tabletag.tld 标签文件，描述自定义的标签及配置自定义标签文件，关键代码如下：

```
<tag>
  <name>tbg</name>
<tag-class>com.TableTag</tag-class>
<body-content>empty</body-content>
<attribute>
<name>tableTitle</name>
<required>true</required>
<rtexprvalue>true</rtexprvalue>
</attribute>
<attribute>
<name>sqlSelect</name>
<required>true</required>
<rtexprvalue>true</rtexprvalue>
</attribute>
</tag>
```

（4）在 web.xml 配置文件中引入 tabletag.tld 标签文件，关键代码如下：

```
<taglib>
<taglib-uri>/tabletag</taglib-uri>
<taglib-location>/WEB-INF/tabletag.tld</taglib-location>
</taglib>
```

（5）创建 index.jsp 页面文件，引入自定义的标签后，先定义两个参数用来为标签属性赋值，然后在这个页面中使用该标签，关键代码如下：

```
<%@ taglib uri="/WEB-INF/tabletag.tld" prefix="tagtable"%>
<%
String[] tableTitle = {"姓名","性别","年龄","身份证号","政治面貌","家庭电话","家庭地址"};
String sqlSelect = "SELECT top 6 name, sex, age, sfzhm, zzmm, jtdh, jtdz FROM docu_stu_info ";
request.setAttribute("title",tableTitle);
request.setAttribute("sql",sqlSelect);
%>
<body>
    <tagtable:tbg tableTitle="${title}" sqlSelect="${sql}"/>
</body>
```

秘笈心法

在实际项目开发过程中，经常需要进行数据查询，而数据查询的实现过程都是类似的，因此可以考虑实现一个公共的自定义查询标签，在查询数据时，调用自定义标签即可，这样可以提高开发效率。

实例 123	自定义生成随机数的标签 光盘位置：光盘\MR\04\123	初级 实用指数：★★

实例说明

JSP 自定义标签可以减少 JSP 页面中嵌入的 Java 脚本，使得页面程序易于维护，并提高代码重用率。本实例使用自定义标签在页面中输出一个用户指定位数的随机数，如图 4.33 所示。

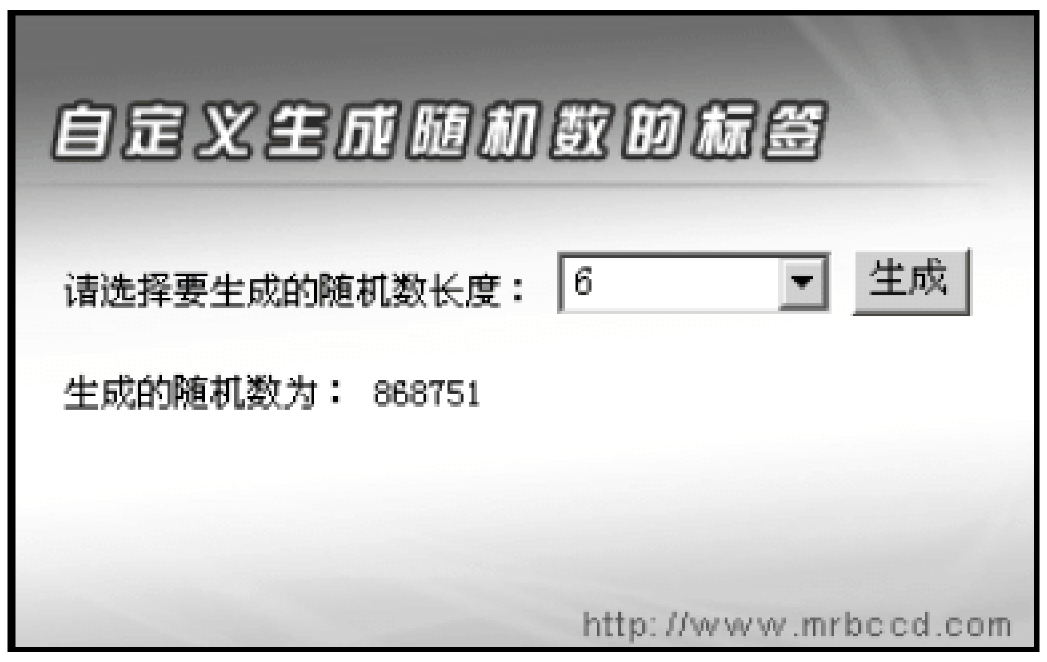


图 4.33　生成随机数

关键技术

本实例中设计的标签含有一个 length 属性，这就需要在标签处理类中定义一个相对应的属性和 get、set 方法。另外，还需要在标签描述文件中配置这个属性。

设计过程

（1）编写 RandomNum.java 类文件，继承自 TagSupport 类，并在该类中编写一个生成随机数的方法 createRandom()，在 doEndTag()方法中调用该方法生成一个随机数，关键代码如下：

```
public class RandomNum extends TagSupport {
 private static final long serialVersionUID = 1L;
 protected String length;                                        //标签属性
 public String getLength() {
         return length;
 }
 public void setLength(String length) {
         this.length = length;
 }
 public int createRandom(){                                      //根据长度生成一个随机数
         int i = (int)(Math.random()*(Math.pow(10, Integer.parseInt(length))));
         return i;
 }
 public int doEndTag() throws JspException {
         try {
                 pageContext.getOut().print(createRandom());     //获取输出对象，输出随机数
         } catch (IOException e) {
                 e.printStackTrace();
         }
         return super.doEndTag();
 }
}
```

（2）在 WEB-INF 文件夹中编写 mytag.tld 描述文件，关键代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<taglib xmlns="http://java.sun.com/xml/ns/j2ee"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee web-jsptaglibrary_2_0.xsd"
      version="2.0">
        <tlib-version>1.0</tlib-version>
        <short-name>mytag</short-name>
        <uri>http://www.tag.com/mytag</uri>
        <tag>
                <description>生成随机数</description><!-- 标签描述 -->
                <name>randomNum</name><!-- 标签名称 -->
        <tag-class>com.jwy.tag.RandomNum</tag-class><!-- 标签路径 -->
        <body-content>empty</body-content><!-- 标签体 -->
        <attribute>
                <name>length</name><!-- 属性名称 -->
                <required>true</required><!-- 是否必须指定此属性 -->
                <rtexprvalue>true</rtexprvalue><!-- 是否能接受 JSP 表达式或 EL 表达式 -->
        </attribute>
        </tag>
</taglib>
```

（3）在 web.xml 中对自定义标签库进行引用，关键代码如下：

```
<jsp-config>
        <taglib>
                <taglib-uri>http://www.tag.com/mytag</taglib-uri>
                <taglib-location>/WEB-INF/mytag.tld</taglib-location>
        </taglib>
</jsp-config>
```

（4）编写 index.jsp 页面，在此页面中使用自定义标签，关键代码如下：

```
<body>
<form action="index.jsp" method="post" name="f1">
        请选择要生成的随机数长度：
        <select name="len">
                <option selected="selected" value="0">请选择位数</option>
                <option value="1">1</option>
                <option value="2">2</option>
                <option value="3">3</option>
                <option value="4">4</option>
                <option value="5">5</option>
                <option value="6">6</option>
                <option value="7">7</option>
                <option value="8">8</option>
                <option value="9">9</option>
                <option value="10">10</option>
        </select>
        <input type="submit" value="生成">
</form>
<%
        if (request.getParameter("len")!=null&&!"0".equals(request.getParameter("len"))) {
%>
生成的随机数为：
<mytag:randomNum length='<%=request.getParameter("len")%>' />
<%
        }
%>
</body>
```

秘笈心法

根据本实例的实现，读者可以使用自定义标签生成验证码。

实例 124　自定义生成系统菜单的标签

光盘位置：光盘\MR\04\124

初级

实用指数：★★

实例说明

本实例使用自定义标签来生成产品分类导航目录。运行本实例，如图 4.34 所示，单击产品类别名称就可以看到该类别中的全部产品。



图 4.34　产品导航目录

关键技术

如果要在页面中使用自定义标签，首先要在 web.xml 文件中对标签库进行引用声明（具体代码见设计过程步骤（4）），然后还要在页面中引用标签，关键代码如下：

```
<%@taglib prefix="mytag" uri="http://www.tag.com/mytag"%>
```

设计过程

（1）编写 MenuTag.java 类文件，并让该类继承 TagSupport 类，然后重写父类中的 doStartTag()方法，并在此方法中调用自定义显示菜单的方法，关键代码如下：

```
public class MenuTag extends TagSupport {
 public int doStartTag() throws JspException {
        HttpServletRequest request = (HttpServletRequest) pageContext.getRequest();     //创建 request 对象
        HttpSession session = request.getSession();                                     //创建 session 对象
        List list = (List) session.getAttribute("menuList");                            //获取会话对象中保存的 List 列表
        loadMenu(list);                                                                 //调用生成菜单的方法
        return super.doStartTag();
 }
}
```

（2）在 MenuTag 类中编写 loadMenu()方法，该方法将 List 列表中保存的菜单内容显示在页面中，并将用户选中的项目展开，关键代码如下：

```
private void loadMenu(List menuList) {
 JspWriter out = pageContext.getOut();                                                  //获取向页面输出的 out 对象
 HttpServletRequest request = (HttpServletRequest) pageContext.getRequest();            //创建 request 对象
 HttpSession session = request.getSession();                                            //创建 session 对象
 try {
        if (menuList.isEmpty()) {
                out.write("<table><tr><td>没有可以显示的菜单</td></tr></table>");
        } else {
                for (int i = 0; i < menuList.size(); i++) {                             //循环 List 列表
                        List sList = (List) menuList.get(i);
                        out.write("<a href='index.jsp?id=" + i + "'>"+ sList.get(0) + "</a><br>");
                        if (pageContext.getRequest().getParameter("id") != null) {      //判断是否被选中
                                if (Integer.parseInt(pageContext.getRequest().getParameter("id")) == i) {
                                        for (int j = 1; j < sList.size(); j++) {        //显示子项目
                                                out.write("  " + sList.get(j) + "<br>");
                                        }
                                }
                        }
                }
        }
 } catch (Exception e) {
        e.printStackTrace();
 }
}
```

（3）编写 mytag.tld 标签描述文件，关键代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<taglib xmlns="http://java.sun.com/xml/ns/j2ee"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee web-jsptaglibrary_2_0.xsd"
      version="2.0">
        <tlib-version>1.0</tlib-version>
        <short-name>mytag</short-name>
        <uri>http://www.tag.com/mytag</uri>
        <tag>
                <description>生成菜单</description>
                <name>nemuTag</name>
        <tag-class>com.jwy.tag.MenuTag</tag-class>
        <body-content>empty</body-content>
        </tag>
</taglib>
```

（4）在 web.xml 配置文件中对标签库进行引用声明，关键代码如下：

```
<jsp-config>
  <taglib>
    <taglib-uri>http://www.tag.com/mytag</taglib-uri>
    <taglib-location>/WEB-INF/mytag.tld</taglib-location>
  </taglib>
</jsp-config>
```

（5）编写 index.jsp 页面文件，首先引入自定义标签库，然后在该文件中初始化一个 List 列表对象并保存到 session 对象中，最后调用该标签，关键代码如下：

```
<%@page contentType="text/html" pageEncoding="GBK"%>
<%@page import="java.util.*"%>
<%@taglib prefix="mytag" uri="http://www.tag.com/mytag"%>
<html>
<body>
        <%
                if (session.getAttribute("menuList") == null) {
                        List<List<String>> menuList = new ArrayList<List<String>>();
                        List<String> sList1 = new ArrayList<String>();
                        sList1.add("硬件");
                        sList1.add("显示器");
                        sList1.add("主机");
                        sList1.add("显卡");
                        sList1.add("CPU");
                        sList1.add("内存");
                        sList1.add("硬盘");
                        menuList.add(sList1);
                        List<String> sList2 = new ArrayList<String>();
                        sList2.add("外设");
                        sList2.add("键盘");
                        sList2.add("鼠标");
                        sList2.add("音箱");
                        sList2.add("耳机");
                        sList2.add("摄像头");
                        menuList.add(sList2);
                        ……//省略部分代码
                        session.setAttribute("menuList", menuList);
                }
        %>
        产品导航目录
        <br>
        <mytag:nemuTag />
</body>
</html>
```

秘笈心法

根据本实例，读者可以编写论坛中的功能菜单，还可以从数据库中读取菜单内容。

第 5 章

JavaBean 技术

- 字符串处理
- 数据验证
- 日期时间处理
- 输出实用的 HTML 代码
- 窗口与对话框
- 对数据库操作的 JavaBean

5.1　字符串处理

在开发 Web 应用程序时，由于大部分数据都是以字符串形式表示的，所以避免不了要对这些字符串数据进行处理。例如，有时需要将小写金额转换为大写；将一个长的字符串进行截取；过滤字符串中的空格等，这些操作在程序中会经常用到。本节将介绍一些常用的字符串处理的 JavaBean。

实例 125　小写金额转换成大写金额

光盘位置：光盘\MR\05\125

高级

实用指数：★★★★

实例说明

运行本实例后，在页面中输入数字格式的金额，然后单击“转换”按钮后会显示金额的大写汉字形式，运行结果如图 5.1 所示。



图 5.1　小写金额转换为大写

关键技术

实现本实例，首先需要在 JavaBean 类中定义一个表示大写数字的数组和一个表示数字位数的数组，代码如下：

```
String numberCNN[]={"零","壹","贰","叁","肆","伍","陆","柒","捌","玖"};
String submoneyCN[]={"","拾","佰","仟"};
```

然后在自定义的 JavaBean 方法中循环处理数字字符串的每一位数字，经过判断将每位数字都转换为大写汉字，最后组合成一个完整的以大写汉字表示的金额字符串。

设计过程

（1）新建名为 StringUtil 的 JavaBean 类，关键代码如下：

```
public class StringUtil {
        private String money;                                              //转换之前的数字金额
        private String submoneyCN[]={"","拾","佰","仟"};                    //表示数字位数的数组
        private String numberCNN[]={"零","壹","贰","叁","肆","伍","陆","柒","捌","玖"};  //大写数字的数组
        public StringUtil(){}                                              //默认的构造方法
                … //此处省略了 formatCN 以及 money 属性的 getXXX()和 setXXX()方法
        /**
         * 转换数字金额为大写金额
         * @return 返回转换后的大写金额
         */
        public String convert(){
                int point=money.indexOf(".");                              //判断字符串是否包含小数点
                if(point!=-1){
                        String money1=money.substring(0,point);            //小数点之前的部分
                        String money1_1=(new StringBuffer(money1).reverse()).toString();  //字符串倒序
                        String money2=money.substring(point+1);            //小数点之后的部分
                        if(money2.length()<2){                             //如果小数点后的部分少于两位，用 0 补齐
                                if(money2.length()==0) money2="00";
                                else money2+="0";
                        }
                        else                                               //如果小数点后的位数大于两位，则只取前两位
                                money2=money.substring(point+1,point+3);
                int len = money1_1.length();                               //字符串反转之后，小数点之前的字符位数
                int pos=len-1;
```

```
            String sigle="";
            boolean allhavenum=false;
            boolean havenum=false;
            boolean    mark=false;                        //若当前数为 0，将该值设为 true，否则设为 false
            /**以下代码为读出小数点左面的部分*/
            while(pos>=0){
               //截取 1 位数字，如数字 1234.56，将左侧的字符串反转，值为 4321，则截取的值为 1
                  sigle=money1_1.substring(pos,pos+1);
                  /**读取“亿单元”的代码
                    *  假设读取 10024531042.34
                    *  小数点左面反转后为 24013542001
                    * pos 的初始值为 10
                    * mark 的初始值为 false
                    * havenum 的初始值为 false
                    * */
                  if(pos>=8&&pos<12){
                        if(!sigle.equals("0")){ //如果当前值不为 0
                              if(!mark){       //如果当前值的前一位数不为 0
                                    formatCN+=numberCNN[Integer.parseInt(sigle)]+submoneyCN[pos%4];
                              }
                              else{     //如果当前值不为 0，但该值的前一位数为 0
                                    //如果在当前值之前有不为 0 的数字出现
                                    //该条件用来处理用户输入的如 0012.34 的数值
                                    if(allhavenum) formatCN+="零";
                                                formatCN+=numberCNN[Integer.parseInt(sigle)]+submoneyCN[pos%4];
                                    mark=false;
                              }
                              havenum=true;
                                //allhavenum 表示小数点左面的数中是否有不为 0 的数字
                              allhavenum=true;
                        }
                        else{ //如果当前值为 0
                              mark=true;
                        }
                          //如果当前数字为该单元最后一位，并且该单元中有不为 0 的数字出现
                        if(pos%4==0&&havenum){
                                formatCN+="亿";
                                havenum=false;
                        }
                  }
                  … //此处省略了其他单元的转换，其转换过程与以上代码“亿单元”的转换过程类似
                  pos--;
            }
            /**碰到小数点时的读法*/
         if(allhavenum) //allhavenum 表示小数点左面的内容中是否有数字出现
                     formatCN+="元";
         else//如果小数点左面的部分都为 0，如 00.34 应读为：零元 3 角 4 分
                     formatCN="零元";
         /**以下代码为读出小数点右面的部分 */
         if(money2.equals("00"))
              formatCN+="整";
         else{
              /**读出角，如 120.34 读为：1 佰 2 拾元零 3 角 4 分；123.04 读为：1 佰 2 拾 3 元零 4 分*/
              if(money2.startsWith("0")||(allhavenum&&money1.endsWith("0"))){
                     formatCN+="零";
             }
              if(!money2.startsWith("0")){
                     formatCN+=numberCNN[Integer.parseInt(money2.substring(0,1))]+"角";
             }
              //读出分，如 12.30 读为：1 拾 2 元 3 角零分
              formatCN+=numberCNN[Integer.parseInt(money2.substring(1))]+"分";
         }
   }
   else{
         formatCN="输入的格式不正确！格式：888.00";
   }
```

```
            return formatCN;
        }
}
```

（2）新建 index.jsp 页面，用于输入数字金额，关键代码如下：

```
<form action="convert.jsp" method="post">
        <table>
                <tr>
                        <td align="center" bgcolor="skyblue"> 请输入金额</td>
                </tr>
                <tr height="25">
                        <td bgcolor="yellow">
                                金额：<input type="text" name="money" id="money" />
                                <input type="submit" value="转  换">
                        </td>
                </tr>
        </table>
</form>
```

（3）新建 convert.jsp 的处理页，用于获得表单的请求信息，并调用 Bean 类的方法转换金额，关键代码如下：

```
<%
        String money=request.getParameter("money");
%>
<jsp:useBean id="strBean" class="com.lh.bean.StringUtil"></jsp:useBean>
<jsp:setProperty property="money" name="strBean" value="<%=money %>"/>
<table>
        <tr><td align="center" bgcolor="skyblue"> 转换结果：</td></tr>
        <tr height="25">
                <td bgcolor="yellow"><jsp:getProperty property="money" name="strBean"/></td>
        </tr>
</table>
```

秘笈心法

本实例的实现关键是如何找到其中的算法规律。例如，读取 123456789.34 这样一个数字，可以将数据以小数点为界限分为两个部分，小数点左侧的数字从个位起每 4 位组成一个单元，然后像读取个、十、百、千的方法读取各个单元，并加上该单元的单位；小数点右侧的数字可以直接读取。

实例 126　转换输入文本中的回车和空格

光盘位置：光盘\MR\05\126

初级

实用指数：★★★☆

实例说明

本实例将用户输入的回车和空格转换成能够在 JSP 页中输出的回车和空格。运行本实例，在输入框中输入一段包含空格和回车的内容，提交表单后，在显示信息页中的内容会包含空格和回车，运行结果如图 5.2 所示。



图 5.2　转换输入文本中的回车和空格

关键技术

本实例主要应用 String 类中的 replaceAll()方法，将用户输入的空格和回车替换为 HTML 代码中的
和 ，该方法用于将字符串中的某个子字符串替换为指定的字符串，其语法格式如下：

```
replaceAll(String regex, String replacement)
```

参数说明

❶ regex：字符串中原来的子字符串。

❷ replacement：替换后的子字符串。

设计过程

（1）新建 StringUtil 的 JavaBean 类，该类主要包含一个转换空格和换行符的方法，关键代码如下：

```
public class StringUtil {
        private String str;                                    //要替换的字符串
        public void setStr(String str){
                this.str = str;
        }
        public String getStr(){
                return replace(str);
        }
        /**
         * 替换字符串的方法
         * @param str：源字符串
         * @return  替换后的字符串
         */
        public String replace(String str){
                String newStr1="";
                String newStr2="";
                newStr1 = str.replaceAll(" ", " ");          //替换字符串中的空格为 
                newStr2 = newStr1.replaceAll("\r\n", "<br>");   //替换换行符为<br>
                return newStr2;                                 //返回替换后的字符串
        }
}
```

（2）新建 index.jsp 页面，用于在表单中输入包含空格和换行的测试内容，关键代码如下：

```
<form action="replace.jsp" method="post">
   <table>
        <tr><td align="center">请输入信息</td>
        </tr>
        <tr>
                <td><textarea rows="5" cols="30" name="info" ></textarea></td>
        </tr>
        <tr>
                <td align="center"><input   type="submit" value="提  交" /></td>
        </tr>
   </table>
</form>
```

（3）新建 replace.jsp 页面，该页面用于获得表单信息，并调用 Bean 类的方法实现字符串中的空格和换行的转换，关键代码如下：

```
<body>
        <%
                String info = request.getParameter("info");
        %>
    <jsp:useBean id="strBean" class="com.lh.bean.StringUtil"></jsp:useBean>
    <jsp:setProperty property="str" name="strBean" value="<%=info %>"/>
    <table width="240" >
                <tr>
                        <td align="center">查看信息结果</td>
                </tr>
                <tr>
                        <td height="100" valign="top">
                                <jsp:getProperty property="str" name="strBean"/>
                        </td>
                </tr>
        </table>
</body>
```

秘笈心法

转换空格和回车经常应用在论坛网站的用户留言板中。例如，在留言板中，用户输入留言内容时，可能会故意输入回车和空格，因此在显示用户的留言内容时，应该包含用户输入的回车和空格。

实例 127　**计算字符串的实际长度**　初级

光盘位置：光盘\MR\05\127　实用指数：★★★★

实例说明

在程序开发过程中，字符串类型的数据使用得非常频繁，经常需要获得字符串的实际长度，然后可以根据该字符串的长度值来操作字符串，本实例的运行结果如图 5.3 所示。

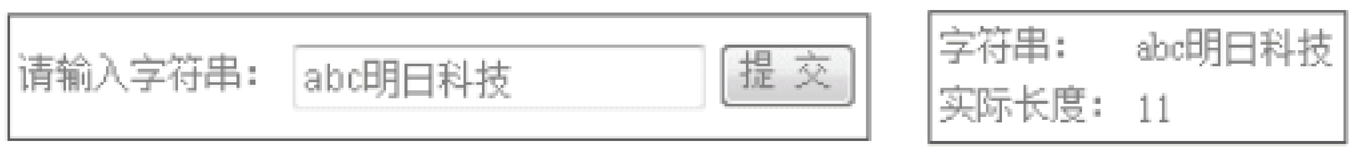


图 5.3　计算字符串的实际长度

关键技术

本实例主要应用 String 类的 getLength 方法、toCharArray 方法和 getBytes 方法。如果字符串中不包含中文字符，那么使用 getLength 方法就可以获得字符串的长度。使用 getLength 方法时，字符串的每一位字符的长度都被认为是 1，由于中文字符占两位长度，所以使用 getLength 计算出的包含中文字符串的长度是不对的。

计算包含中文字符串的长度时，可以配合使用 toCharArray 方法和 getBytes 方法。首先使用 toCharArray 方法将一个字符串转换为字符数组，然后循环这个字符数组，将每个字符通过 getBytes 方法转换为字节数组并获取字节数组长度，最后累计每个字符的长度，从而获得整个字符串的实际长度。

设计过程

（1）新建名为 StringUtil 的 JavaBean 类，该类中提供的方法用于获得字符串的实际长度，关键代码如下：

```
public class StringUtil {
	private String str;                                   //需要计算长度的字符串
	private int strLength;                                //字符串的实际长度
	public String getStr() {
		return str;
	}
	public void setStr(String str) {
		this.str = str;
	}
	public int getStrLength() {
		char[] c =str.toCharArray();                      //将字符串转换为字符数组
		int factualLength = 0;                            //用于保存每个字符的实际长度
		for(int i=0;i<c.length;i++){
			factualLength =String.valueOf(c[i]).getBytes().length;   //获得字节数组的长度
			if(factualLength==3){                         //当程序编码为 UTF-8 时，汉字实际的字节长度为 3
				factualLength =2;                         //此处将字节长度改为 2
			}
			strLength+=factualLength;                     //将每个字符的长度累加，结果就是字符串的总长度
		}
		return strLength;
	}
	public void setStrLength(int strLength) {
		this.strLength = strLength;
	}
}
```

（2）新建 index.jsp 网页，该页中包含一个提交字符串信息的表单，关键代码如下：

```
<form action="getlength.jsp" method="post">
	<table>
		<tr height="35">
			<td align="center">请输入字符串：</td>
			<td>
				<input type="text" name = "str" />
			</td>
			<td align="center">
```

```
                <input  type="submit" value="提 交" />
            </td>
        </tr>
    </table>
</form>
```

（3）新建 getlength.jsp 页，在该页中首先获得表单中的字符串，然后调用 StringUtil 类的方法计算字符串的长度，关键代码如下：

```
<body>
      <%
          String str = request.getParameter("str");
      %>
      <jsp:useBean id="strBean" class="com.lh.bean.StringUtil"></jsp:useBean>
      <jsp:setProperty property="str" name="strBean" value="<%=str %>"/>
    <table>
      <tr><td>字符串：</td>
          <td align="left"><jsp:getProperty property="str" name="strBean"/></td>
      </tr>
      <tr><td>实际长度：</td>
          <td><jsp:getProperty property="strLength" name="strBean"/></td>
      </tr>
      </table>
  </body>
```

秘笈心法

由于在数据库中保存的字符串类型的值是按实际的字符串长度保存的，英文占一个长度，而汉字字符占两个长度。如果保存到数据库的字符串长度超出数据库中定义的实际长度，将会抛出异常导致程序终止。因此在将数据保存到数据库之前，判断字符串的实际长度是很有必要的。

实例 128 字符串截取

光盘位置：光盘\MR\05\128

初级

实用指数：★★★★

实例说明

在实际程序开发中，经常会对字符串进行截取，如在论坛网站的帖子列表中，当某个帖子的标题过长时，需要截取该帖子标题的部分内容显示在列表中，然后其余内容以省略号代替。运行本实例，效果如图 5.4 所示。

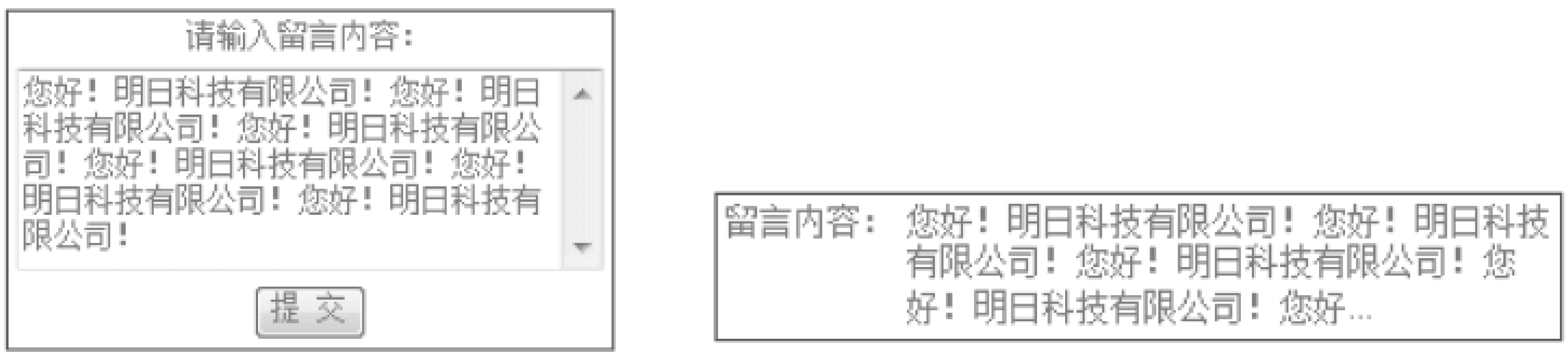


图 5.4 字符串截取

关键技术

本实例主要使用 String 类中的 substring()方法实现字符串截取，其语法格式如下：

```
substring(int beginIndex,int endIndex)
```

参数说明

❶ beginIndex：指定截取字符串的起始位置。

❷ endIndex：指定截取字符串的结束位置。该参数为可选参数，如果不指定此参数，则截取时是从起始位置 beginIndex 截取到字符串的末尾。

设计过程

（1）新建名为 StringUtil 的 JavaBean 类，在该类中实现字符串的截取操作，关键代码如下：

```
public class StringUtil {
        private String str;              //需要截取的源字符串
        public String getStr() {
                if(str.length()>50){   //如果字符串的长度大于 50，则从 0 开始截取到 50，之后的以省略号代替
                        return str.substring(0,50)+"...";
                }
                return str;
        }
        public void setStr(String str) {
                this.str = str;
        }
}
```

（2）新建 index.jsp 页，该页包含一个输入留言内容的表单，关键代码如下：

```
<form action="substr.jsp" method="post">
  <table>
        <tr>
                <td align="center">请输入留言内容：</td></tr>
        <tr>
                <td><textarea rows="5" cols="30" name="str"></textarea></td>
        </tr>
        <tr>
                <td align="center"><input   type="submit" value="提  交" /></td>
        </tr>
  </table>
</form>
```

（3）新建 substr.jsp 页，用于处理表单提交过来的留言信息，关键代码如下：

```
<%
        String str = request.getParameter("str");
%>
<jsp:useBean id="strBean" class="com.lh.bean.StringUtil"></jsp:useBean>
<jsp:setProperty property="str" name="strBean" value="<%=str %>"/>
<table width="200">
        <tr>
                <td width="60" valign="top">留言内容：</td>
                <td align="left"><jsp:getProperty property="str" name="strBean"/></td>
        </tr>
</table>
```

秘笈心法

在 StringBuffer 类中同样包含 substring()方法，该方法也可以实现字符串截取。StringBuffer 用于表示可以修改的字符串，StringBuffer 是与 String 同等的类，它表示可增加和可编写的字符序列。

实例 129 字符串转换成数组

初级

光盘位置：光盘\MR\05\129

实用指数：★★★★

实例说明

将字符串转换为数组也是在开发中常用的技术，如用户在某个表单页中选中了多个复选框，在处理这些选中的内容时，需要获得复选框的所有选中内容，此时需要使用 JavaScript 方法将所有选中的内容累加，然后将这个累加后的字符串提交，在服务器端获得该字符串时，就需要将这个包含多个复选框内容的长字符串分解为数组，然后再进一步处理。运行本实例，在“您喜欢的运动有：”中，选中多个复选框内容，然后提交表单，最后在服务器处理页中将复选框的内容转换为数组并显示，运行结果如图 5.5 所示。



图 5.5　字符串转换为数组

关键技术

本实例主要通过 String 类的 split()方法来实现将字符串转换为数组。split()方法包含一个 String 类型的参数 regex，调用时会以 regex 为字符串的分隔符，将字符串分隔为字符串数组，代码如下：

```
String str = "a,b,c,d";
String strArr[] = str.split(",");
```

设计过程

（1）新建 StringUtil 的 JavaBean 类，在该类中实现将字符串转换为数组，关键代码如下：

```
public class StringUtil {
        private String str;                //要分隔的字符串
        private String strArr[];           //分隔后的字符串数组
        private String listSeparator;      //分隔符号
        public StringUtil(){}              //默认构造方法
        public String getStr() {
                return str;
        }
        public void setStr(String str) {
                this.str = str;
        }
        /**返回分隔符*/
        public String getListSeparator() {
                return listSeparator;
        }
        /**设置分隔符*/
        public void setListSeparator(String listSeparator) {
                this.listSeparator = listSeparator;
        }
        /**返回字符串数组*/
        public String[] getStrArr() {
                return str.split(listSeparator); //根据分隔符号分隔字符串为数组
        }
}
```

（2）新建 index.jsp 页，在该页中主要包含一个隐藏域和多个复选框的表单，关键代码如下：

```
<form action="toarray.jsp" method="post">
        <input type="hidden" name = "likes" id="likes" />
        <table width="220">
                <tr bgcolor="skyblue"><td align="center">您喜欢的运动有：  </td></tr>
                <tr>
                        <td>
                                <input type="checkbox" name="like" value="打篮球">打篮球
                                <input type="checkbox" name="like" value="踢足球">踢足球
                                <input type="checkbox" name="like" value="打乒乓球">打乒乓球
                                <input type="checkbox" name="like" value="跑步">跑步
                                <input type="checkbox" name="like" value="打羽毛球">打羽毛球
                                <input type="checkbox" name="like" value="游泳">游泳
                        </td>
                </tr>
                <tr bgcolor="skyblue">
                  <td align="center">
                        <input   type="submit" value="提 交" onclick="getSelectCheckbox()" />
                  </td>
                </tr>
        </table>
</form>
```

（3）编写“提交”按钮的 onClick 事件所调用的 JavaScript 方法，该方法用于获得所有选中复选框的内容，并将该值赋给隐藏域，关键代码如下：

```
<script type="text/javascript">
        function getSelectCheckbox(){
                var checkObj = document.getElementsByName("like");        //获得复选框对象，该对象是个数组
                var likeStr ="";
                for(var i=0;i < checkObj.length ;i++){
```

```
                if(checkObj[i].checked==true){                  //如果某个复选框为选中状态
                    likeStr+=checkObj[i].value+",";             //将选中的复选框值组合，并以“,”分隔
                }
            }
            //将组合后复选框的内容赋给隐藏域，表单提交后，获得的是该隐藏域的内容
            document.getElementById("likes").value = likeStr ;
        }
</script>
```

（4）新建 toarray.jsp 页，该页用于处理表单请求，并调用 JavaBean 方法将字符串转换为数组，然后将数组中的元素显示在表格中，关键代码如下：

```
<%
    String likes = request.getParameter("likes");               //从请求中获取以“,”分隔的字符串
%>
<!-- 使用 useBean 动作标签导入 JavaBean 对象 -->
<jsp:useBean id="strBean" class="com.lh.bean.StringUtil"></jsp:useBean>
<!-- 对 StringUtil 类的对象 strBean 的 str 属性赋值 -->
<jsp:setProperty property="str" name="strBean" value="<%=likes %>"/>
<!-- 对 StringUtil 类的对象 strBean 的 listSeparator（分隔符）属性赋值 -->
<jsp:setProperty property="listSeparator" name="strBean" value=","/>
<%
    String likeArr[] = strBean.getStrArr();                     //分隔后的字符串数组
%>
<table>
    <tr>
        <td width="100" valign="top">您选择的运动有：</td>
        <%
            for(int i = 0;i<likeArr.length;i++){
         %>
        <td align="left">【<%=likeArr[i] %>】</td>
        <%} %>
    </tr>
</table>
```

秘笈心法

将字符串转换成数组的操作在实际的程序开发中经常用到，所以读者应该掌握。

实例 130	数组转换为字符串 光盘位置：光盘\MR\05\130	初级 实用指数：★★★☆

实例说明

在实际的开发过程中，经常需要将数组转换为字符串。本实例将介绍如何将数组元素转换为字符串并显示，运行结果如图 5.6 所示。



图 5.6　将月份数组转换为字符串显示

关键技术

将数组元素转换为字符串非常简单，主要通过循环数组，然后在循环中依次将每个元素取出来，再把所有元素拼接成一个字符串。

设计过程

（1）新建名为 StringUtil 的 JavaBean 类，该类主要用于将数组转换为字符串，关键代码如下：

```
public class StringUtil {
    private int intArr[];                                       //整型数组
    private String str="";                                      //将数组转换后的字符串
    public StringUtil(){}                                       //默认构造方法
    /**返回字符串*/
    public String getStr() {
        for(int i=0;i<intArr.length;i++){                       //循环整型数组
```

```
                    str+=String.valueOf(intArr[i]);          //将每个数组元素转换为字符串并拼成字符串
                    if(i<intArr.length-1){
                            str+=",";                        //每个字符串以“,”隔开
                    }
            }
            return str;
    }
    /**给字符串赋值*/
    public void setStr(String str) {
            this.str = str;
    }
    /**返回数组*/
    public int[] getIntArr() {
            return intArr;
    }
    /**给数组赋值*/
    public void setIntArr(int[] intArr) {
            this.intArr = intArr;
    }
}
```

（2）新建 index.jsp 网页，在该页中定义一个整型的表示月份的数组，然后调用 StringUtil 类中的方法将数组转换为字符串并显示在表格中，关键代码如下：

```
<%
    int month[] =new int[12];                            //创建一个月份的数组
    for(int i=0;i<month.length;i++){                     //通过循环给月份数组的元素赋值
            month[i] = i+1;
    }
%>
<!-- 使用 useBean 动作导入 StringUtil 类 -->
<jsp:useBean id="strBean" class="com.lh.bean.StringUtil"></jsp:useBean>
<!-- 对 StringUtil 类对象中的 intArr 数组属性赋值 -->
<jsp:setProperty property="intArr" name="strBean" value="<%=month %>"/>
<table width="220">
        <tr bgcolor="skyblue">
                <td align="center">一年中的月份：</td></tr>
        <tr>
                <td align="center">
                        <!-- 获得 StringUtil 类对象中的 str 属性值，该值为数组转换后的字符串 -->
                        <jsp:getProperty property="str" name="strBean"/>
                </td>
        </tr>
</table>
```

秘笈心法

将数组元素转换为字符串的操作在实际的程序开发中经常用到，所以读者应该掌握转换的方法。

实例 131　将整型值转换为字符串

光盘位置：光盘\MR\05\131

初级

实用指数：★★★☆

实例说明

在实际的开发过程中，经常需要将整型值转换为字符串。本实例将通过几种方法来介绍如何将整型值转换为字符串型，运行结果如图 5.7 所示。

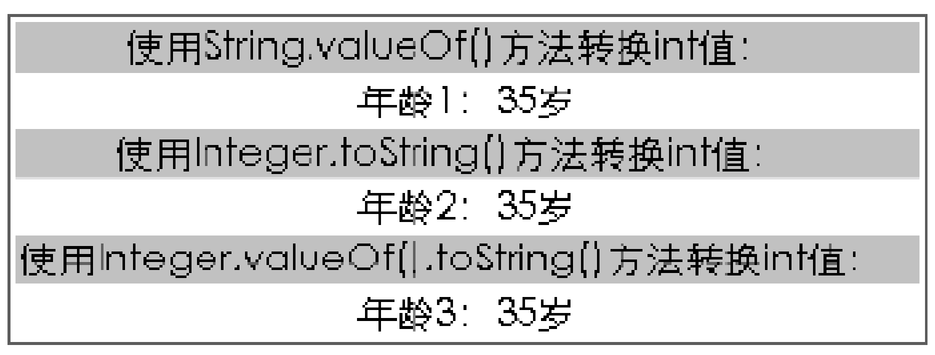


图 5.7　将整型年龄转换为字符串

关键技术

将整型值转换为字符串，可以使用以下 3 种简便的方法。

（1）String.valueOf()

String 类的 valueOf()方法实现了将多种类型的值转换为字符串。在 String 类中该方法实现了重载，可以转换的类型包括 boolean 类型的值、char 类型的值、char[]类型的值、double 类型的值、float 类型的值、int 类型的值、long 类型的值以及 Object 类型的值。

（2）Integer.toString()

Integer 类的 toString()方法中包含一个 int 类型的参数，当使用该方法时，可以直接调用该方法并传入一个 int 类型的参数，即可将 int 值转换为字符串。

（3）Integer.valueOf().toString()

使用 Integer 类的 valueOf().toString()方法时，首先将 int 值转换为 Integer 对象，然后再调用 toString()方法即可转换为字符串。

设计过程

（1）新建名为 StringUtil 的 JavaBean 类，在该类中实现了将 int 值转换为字符串的 3 种方法，关键代码如下：

```
public class StringUtil {
 private int intValue=0;                          //转换前的 int 值
 private String strValue1;                        //转换后的字符串 1 值
 private String strValue2;                        //转换后的字符串 2 值
 private String strValue3;                        //转换后的字符串 3 值
 public StringUtil(){}                            //默认构造方法
 public String getStrValue1() {
        return String.valueOf(intValue);          //使用 String 类的 valueOf()方法转换
 }
 public String getStrValue2() {
        return Integer.toString(intValue);        //使用 Integer 类的 toString()方法转换
 }
 public String getStrValue3() {
        return Integer.valueOf(intValue).toString(); //使用 Integer.valueOf().toString()方法
 }
        ……//此处省略了其他属性的 set 方法和 get 方法
}
```

（2）新建 index.jsp 页，在该页中分别调用 StringUtil 类的这 3 种方法返回字符串值，并显示在页面中，关键代码如下：

```
<%
        int userAge = 35;
%>
<!-- 使用 useBean 动作导入 StringUtil 类 -->
<jsp:useBean id="strBean" class="com.lh.bean.StringUtil"></jsp:useBean>
<!-- 对 StringUtil 类对象中的 intValue 属性赋值 -->
<jsp:setProperty property="intValue" name="strBean" value="<%=userAge %>"/>
<table>
        <tr bgcolor="skyblue">
                <td align="center">使用 String.valueOf()方法转换 int 值：</td>
        </tr>
        <tr>
                <td align="center">
                        <!-- 获得 StringUtil 类对象中的 strValue1 属性值 -->
                        年龄 1：<jsp:getProperty property="strValue1" name="strBean"/>岁
                </td>
        </tr>
        <tr bgcolor="skyblue">
                <td align="center">使用 Integer.toString()方法转换 int 值：</td>
        </tr>
        <tr>
                <td align="center">
                        <!-- 获得 StringUtil 类对象中的 strValue2 属性值 -->
```

```
			年龄 2：<jsp:getProperty property="strValue2" name="strBean"/>岁
		</td>
	</tr>
	<tr bgcolor="skyblue">
		<td align="center">使用 Integer.valueOf().toString()方法转换 int 值：</td>
	</tr>
	<tr>
		<td align="center">
			<!-- 获得 StringUtil 类对象中的 strValue3 属性值 -->
			年龄 3：<jsp:getProperty property="strValue3" name="strBean"/>岁
		</td>
	</tr>
</table>
```

秘笈心法

在实际开发过程中，经常需要将整型值转换为字符串类型，因此读者应该至少掌握本实例中讲解的 3 种转换方法中的一种。

实例 132	将字符串型转换为整型 光盘位置：光盘\MR\05\132	初级 实用指数：★★★☆

实例说明

在实际的开发过程中，有时需要将字符串值转换为整型，如用户填写注册表单时输入的年龄或者商品信息表单中的商品数量。本实例将介绍如何把一个字符串值转换为整型，运行结果如图 5.8 所示。

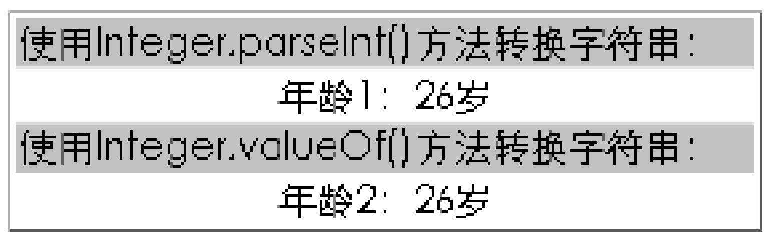


图 5.8　字符串值转换为整型值

关键技术

将字符串转换为整型值，主要使用的是 Integer 类的 parseInt()方法和 valueOf()方法。它们都可以接收一个 String 类型的参数。parseInt()方法返回一个 int 类型的值，valueOf()方法则返回的是 Integer 对象，该对象可以直接作为 int 值输出，也可以通过 intValue()方法转换为 int 类型。当参数不符合转换条件时，如将字符串 a 转换为 int 类型时，会抛出一个 java.lang.NumberFormatException 类型的异常。

设计过程

（1）新建名为 StringUtil 的 JavaBean 类，该类主要实现了将字符串转换为整型的两种方法，关键代码如下：

```
public class StringUtil {
	private int intValue1;			//第一种方法转换后的 int 值
	private int intValue2;			//第二种方法转换后的 int 值
	private String strValue;			//被转换的字符串
	public StringUtil(){}			//默认构造方法
	public int getIntValue1() {
		return Integer.parseInt(strValue);		//使用 Integer 类的 parseInt()方法转换
	}
	public int getIntValue2() {
		return Integer.valueOf(strValue);		//使用 Integer 类的 valueOf()方法转换
	}
	public String getStrValue() {
		return strValue;
	}
	public void setStrValue(String strValue) {
		this.strValue = strValue;
	}
}
```

（2）新建 index.jsp 页，在该页中调用 StringUtil 的两种转换方法，将字符串转换为整型值，关键代码如下：

```
<%
        String strAge = "26";
%>
<!-- 使用 useBean 动作导入 StringUtil 类 -->
<jsp:useBean id="strBean" class="com.lh.bean.StringUtil"></jsp:useBean>
<!-- 对 StringUtil 类对象中的 strValue 属性赋值 -->
<jsp:setProperty property="strValue" name="strBean" value="<%=strAge %>"/>
<table>
        <tr bgcolor="skyblue">
                <td align="center">使用 Integer.parseInt()方法转换字符串：</td>
        </tr>
        <tr>
                <td align="center">
                        <!-- 获得 StringUtil 类对象中的 intValue1 属性值 -->
                        年龄 1：<jsp:getProperty property="intValue1" name="strBean"/>岁
                </td>
        </tr>
        <tr bgcolor="skyblue">
                <td align="center">使用 Integer.valueOf()方法转换字符串：</td>
        </tr>
        <tr>
                <td align="center">
                        <!-- 获得 StringUtil 类对象中的 intValue2 属性值 -->
                        年龄 2：<jsp:getProperty property="intValue2" name="strBean"/>岁
                </td>
        </tr>
</table>
```

秘笈心法

在实际开发过程中，经常需要将字符串转换为整型。如用户注册表单中的年龄，通过表单提交默认为字符串类型，因此在服务器端获取到年龄信息时，就需要将其转换为整型。

实例 133　把整型数据格式化为指定长度的字符串

光盘位置：光盘\MR\05\133

初级

实用指数：★★★☆

实例说明

本实例介绍如何将整型数据格式化为指定长度的字符串，如果位数不够，以 0 补齐，实例运行结果如图 5.9 所示。



图 5.9　将整型值转换为指定长度的字符串

关键技术

实现数字格式化，主要应用的是 java.text.NumberFormat 类，它提供了一些用于将 Number 对象和数字格式化为本地字符串或者通过语义分析把本地化的字符串转换为 Number 对象的方法。本实例用到的几个方法介绍如下。

（1）NumberFormat 类的 getInstance()方法

功能：该方法用于返回当前默认语言环境的通用数值格式。

（2）NumberFormat 类的 setMinimumIntegerDigits()方法

功能：该方法用于设置数的整数部分所允许的最小位数，如果位数不够，则在数字前用 0 补齐。

（3）NumberFormat 类的 format()方法

功能：该方法用于将数字格式化为字符串。

设计过程

（1）新建名为 StringUtil 的 JavaBean 类，该类主要实现了将数字格式化的方法，关键代码如下：

```
public class StringUtil {
 private int intValue;                          //将要格式化的整型值
 private String formatStr;                      //格式化后的字符串
 private int minimumDigit;                      //格式化后字符串的最少位数
 public StringUtil(){}                          //默认的构造方法
 public int getMinimumDigit() {
         return minimumDigit;
 }
 public void setMinimumDigit(int minimumDigit) {
         this.minimumDigit = minimumDigit;
 }
 public int getIntValue() {
         return intValue;
 }
 public void setIntValue(int intValue) {
         this.intValue = intValue;
 }
 public String getFormatStr() {
         NumberFormat nf =NumberFormat.getInstance();       //获取常规数值格式对象
         nf.setMinimumIntegerDigits(minimumDigit);          //设置格式化数字的整数部分最少位数
         return nf.format(intValue).replace(",", "");       //返回格式化的字符串并把字符串中的“,”替换掉
 }
 public void setFormatStr(String formatStr) {
         this.formatStr = formatStr;
 }
}
```

（2）新建 index.jsp 页，该页主要包含一个表单，关键代码如下：

```
<form action="format.jsp" method="post">
       <table>
        <tr>
               <td align="right">请输入要格式化的数字：</td>
               <td><input type="text" name="num" /></td>
        </tr>
        <tr>
               <td align="right">请输入格式化后的字符串长度：</td>
               <td><input type="text" name="length" /></td>
        </tr>
        <tr>
               <td colspan="2" align="center"><input type="submit" value="格式化" /></td>
        </tr>
       </table>
</form>
```

（3）新建 format.jsp 的处理页，用于获得表单请求信息，并调用 StringUtil 类的方法格式化数字，关键代码如下：

```
<%
        String num = request.getParameter("num");           //获取表单中字符串格式的数字
        String length = request.getParameter("length");     //获取字符串格式的长度
        int n = Integer.parseInt(num);                      //转换为 int 类型
        int l = Integer.parseInt(length);
%>
<!-- 使用 useBean 动作标签导入 JavaBean 对象 -->
<jsp:useBean id="strBean" class="com.lh.bean.StringUtil"></jsp:useBean>
<!-- 对 StringUtil 类的 intValue 属性赋值 -->
<jsp:setProperty property="intValue" name="strBean" value="<%=n %>"/>
<!-- 对 StringUtil 类的 minimumDigit 属性赋值 -->
<jsp:setProperty property="minimumDigit" name="strBean" value="<%=l %>"/>
<table>
        <tr>
               <td>格式化之前的数字：</td>
               <td align="left">
                   <!—获得 StringUtil 类的 intValue 属性值 -->
                       <jsp:getProperty property="intValue" name="strBean"/>
```

```
            </td>
        </tr>
        <tr>
            <td>格式化之后的字符串：</td>
            td align="left">
                <!—获得 StringUtil 类的 formatStr 属性值 -->
                <jsp:getProperty property="formatStr" name="strBean"/>
            </td>

        </tr>
</table>
```

秘笈心法

实现本实例可以不用 NumberFormat 类，通过+连接符就可以实现，这一部分代码比较简单，此处不再详细介绍。

实例 134 将长整型的数字分位显示

光盘位置：光盘\MR\05\134

初级

实用指数：★★★☆

实例说明

在实际的开发过程中，有时会遇到显示长整型数字的问题，为了方便用户查看，需要将数字分位显示。本实例将介绍如何将一个长整型的数字分位显示，运行效果如图 5.10 所示。

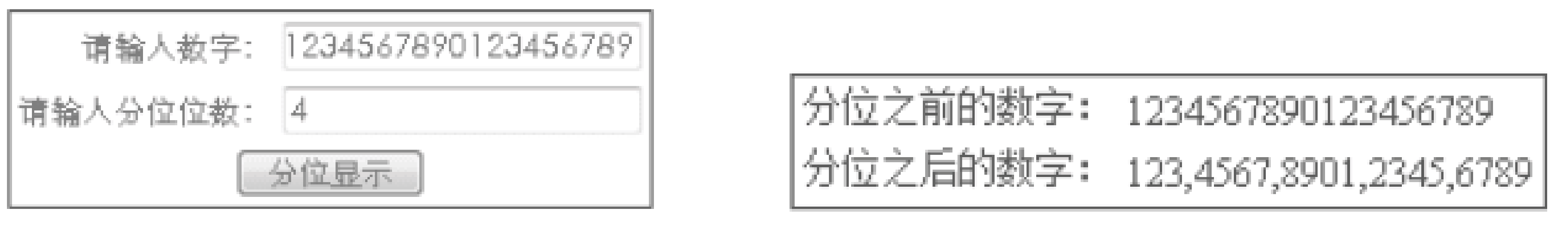


图 5.10　分位显示长数字

关键技术

长整型的数字可以用 long 类型表示，在 Java 中的 long 类型最大值为 19 位的数字 922,3372,0368,5477,5807，定义时应该在数字末尾加上 L，否则会被默认为 int 类型，代码如下：

```
long value = 1234567890123456789l;
```

可以应用 StringBuffer 类的 insert()方法和 reverse()方法将一个 long 类型的数字分位。使用 insert()方法可以动态地向一个 StringBuffer 对象类型的字符串中的指定位置插入指定的字符串，由于分位是从数字的个位开始，所以需要使用 reverse()方法将字符串反转。insert()方法的语法格式如下：

```
public StringBuffer insert(int offset,String str)
```

参数说明

❶ offset：表示偏移量，该参数值必须大于等于 0，且小于等于此序列的长度。

❷ str：表示要插入的字符串。

设计过程

（1）新建 StringUtil 的 JavaBean 类，在该类中实现了将 long 类型的数字转换为分位字符串，关键代码如下：

```
public class StringUtil {
private long longValue;              //要分位的数字
private int digit;                   //分位位数
private String formatStr;            //分位后的字符串
public StringUtil(){}                //默认的构造方法
public long getLongValue() {
        return longValue;
}
public void setLongValue(long longValue) {
        this.longValue = longValue;
}
```

```
public int getDigit() {
        return digit;
}
public void setDigit(int digit) {
        this.digit = digit;
}
public String getFormatStr() {
        StringBuffer sb = new StringBuffer(String.valueOf(longValue));      //将 long 类型的值转换为可动态修改的 StringBuffer 对象
        sb =sb.reverse();                                                   //将字符串反转
        int l = sb.length();
        if(digit==0){                                                       //如果分位位数为 0，设置字符串的长度为分位位数
                digit=l;
        }
        int count = 0;
        /**计算出插入的分位符个数*/
        if(l%digit==0)
                count=l/digit-1;
        else
                count=l/digit;
        for(int i= 0;i<count;i++){
                sb =sb.insert((i+1)*digit+i ,",");                          //在分位的位置插入分位符
        }
        return sb.reverse().toString();
}
public void setFormatStr(String formatStr) {
        this.formatStr = formatStr;
}
}
```

（2）新建 index.jsp 页，该页主要用于输入表单信息，关键代码如下：

```
<form action="format.jsp" method="post">
       <table>
        <tr>
                <td align="right">请输入数字：</td>
                <td><input type="text" name="longValue" /></td>
        </tr>
        <tr>
                <td align="right">请输入分位位数：</td>
                <td><input type="text" name="digit" /></td>
        </tr>
        <tr>
                <td colspan="2" align="center"><input type="submit" value="分位显示" /></td>
        </tr>
       </table>
</form>
```

（3）新建 format.jsp 页，该页用于获取表单信息并调用 StringUtil 类的方法实现长数字的分位，关键代码如下：

```
<%
       String longValueStr = request.getParameter("longValue");            //获取表单中的长数字的字符串
        String digit = request.getParameter("digit");                      //获取分位位数字符串
        long longValue = Long.parseLong(longValueStr);                     //将长数字的字符串转换为 long 类型
        int d = Integer.parseInt(digit);                                   //将分位位数字符串转换为 int 类型
%>
<!-- 使用 useBean 动作标签导入 JavaBean 对象 -->
<jsp:useBean id="strBean" class="com.lh.bean.StringUtil"></jsp:useBean>
<!-- 对 StringUtil 类的 longValue 属性赋值 -->
<jsp:setProperty property="longValue" name="strBean" value="<%=longValue %>"/>
<!-- 对 StringUtil 类的 digit 属性赋值 -->
<jsp:setProperty property="digit" name="strBean" value="<%=d %>"/>
<table>
        <tr >
                <td >分位之前的数字：</td>
                <td align="left">
                        <jsp:getProperty property="longValue" name="strBean"/>
                </td>
        </tr>
        <tr >
                <td >分位之后的数字：</td>
                <td align="left">
```

```
                <jsp:getProperty property="formatStr" name="strBean"/>
            </td>

        </tr>
</table>
```

秘笈心法

对于本实例中计算分位位置的表达式“(i+1)*digit+i”可以这样理解，如果准备插入第一个分隔符即 i=0，那么应该跳过 digit 个字符在第 digit 位置处插入分隔符；如果已经插入了一个分隔符，准备插入第二个分隔符时，则应该跳过(i+1)*digit 个字符，因为在这之前已经插入了一个分隔符，整个长度增加了一位，所以此时插入分隔符的位置应再加上分隔符的个数，即(i+1)*digit+i。

实例 135 过滤输入字符串中的危险字符

初级

光盘位置：光盘\MR\05\135

实用指数：★★★☆

实例说明

在开发网站时，为了避免网站遭到 SQL 语句的注入式攻击，应该考虑到过滤字符串中的危险字符。运行本实例，当在文本框中输入“& ; ' < > -- / % = #”等字符时，在处理页中会把这些字符过滤掉，然后显示出过滤后的字符串，运行效果如图 5.11 所示。

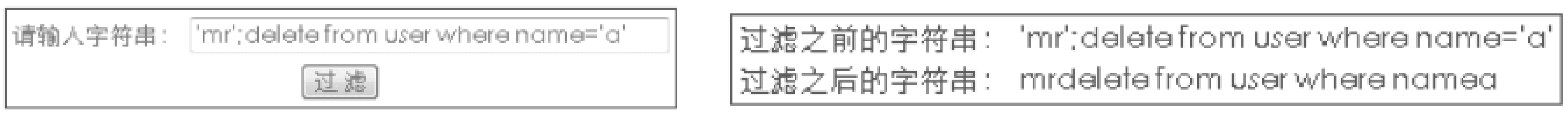


图 5.11　过滤输入字符串中的危险字符

关键技术

本实例主要应用 String 类提供的 replaceAll()方法，该方法用于过滤字符串中包含的所有指定的子字符串，其语法格式如下：

```
public String replaceAll(String regex,String replacement)
```

参数说明

❶ regex：表示需要替换的字符串。

❷ replacement：表示替换后的字符串。

设计过程

（1）新建名为 StringUtil 的 JavaBean 类，该类实现了过滤字符串中的危险字符的方法，关键代码如下：

```
public class StringUtil {
 private String sourceStr;    //源字符串
 private String targetStr;    //替换后的字符串
 public String getSourceStr() {
        return sourceStr;
 }
 public void setSourceStr(String sourceStr) {
        this.sourceStr = sourceStr;
 }
 public String getTargetStr() {
        sourceStr = sourceStr.replaceAll("&", "&amp;");         //过滤字符&
        sourceStr = sourceStr.replaceAll(";", "");              //过滤字符;
        sourceStr = sourceStr.replaceAll("'", "");              //过滤字符'
        sourceStr = sourceStr.replaceAll("<", "&lt;");          //过滤字符<
        sourceStr = sourceStr.replaceAll(">", "&gt");           //过滤字符>
        sourceStr = sourceStr.replaceAll("/", "");              //过滤字符/
        sourceStr = sourceStr.replaceAll("%", "");              //过滤字符%
        sourceStr = sourceStr.replaceAll("=", "");              //过滤字符=
        targetStr = sourceStr;
```

```
        return targetStr;
}
public void setTargetStr(String targetStr) {
        this.targetStr = targetStr;
}
}
```

（2）新建 index.jsp 页，用于输入表单信息，关键代码如下：

```
<form action="filterstr.jsp" method="post">
        <table>
          <tr>
                <td align="right">请输入字符串：</td>
                <td><input type="text" name="sourceStr"    size="40"/></td>
          </tr>
          <tr>
                <td colspan="2" align="center"><input type="submit" value="过  滤" /></td>
          </tr>
        </table>
</form>
```

（3）新建 filterstr.jsp 页，用于获得提交过来的表单信息，并通过 StringUtil 类来实现过滤危险字符，关键代码如下：

```
<%
        String sourceStr = request.getParameter("sourceStr");
%>
<!-- 使用 useBean 动作标签导入 JavaBean 对象 -->
<jsp:useBean id="strBean" class="com.lh.bean.StringUtil"></jsp:useBean>
<!-- 对 StringUtil 类的 sourceStr 属性赋值 -->
<jsp:setProperty property="sourceStr" name="strBean" value="<%=sourceStr %>"/>
<table>
        <tr>
                <td>过滤之前的字符串：</td>
                <td align="left">
                        <jsp:getProperty property="sourceStr" name="strBean"/>
                </td>
        </tr>
        <tr >
                <td>过滤之后的字符串：</td>
                <td align="left">
                        <jsp:getProperty property="targetStr" name="strBean"/>
                </td>
        </tr>
</table>
```

秘笈心法

在开发网站的用户登录模块、信息添加模块以及信息修改模块时，为了防止 SQL 语句的注入式攻击，应该对表单中的数据进行特殊字符过滤。

实例 136	过滤字符串中的空格与 NULL 值 光盘位置：光盘\MR\05\136	初级 实用指数：★★★★☆

实例说明

在用户输入表单信息时，有些文本框的内容不允许为空，而且输入的字符串不允许包含空格。在读取数据库中的数据时，如果有些字段被允许为空值，那么在读取该字段值时将导致程序出现异常，所以在程序中应该过滤掉这些空值或者包含空格的字符串，实例运行效果如图 5.12 所示。



图 5.12　过滤字符串中的空格或 NULL 值

关键技术

本实例主要应用 String 类中提供的 replaceAll()方法，该方法用于替换字符串中指定的子字符串。

设计过程

（1）新建名为 StringUtil 的 JavaBean 类，该类用于过滤字符串中的空格或 NULL 值，关键代码如下：

```
public class StringUtil {
	private String sourceStr;				//源字符串
	private String targetStr;				//过滤后的字符串
	public StringUtil(){}				//默认的构造方法
	public String getSourceStr() {
		return sourceStr;
	}
	public void setSourceStr(String sourceStr) {
		this.sourceStr = sourceStr;
	}
	public String getTargetStr() {
		if(sourceStr==null){				//如果源字符串为 NULL
			sourceStr="";
		}
		sourceStr = sourceStr.replaceAll(" ", "");		//如果源字符串中包含空格则替换
		targetStr = sourceStr;
		return targetStr;
	}
	public void setTargetStr(String targetStr) {
		this.targetStr = targetStr;
	}
}
```

（2）新建 index.jsp 页，用于输入字符串信息，关键代码如下：

```
<form action="filterstr.jsp" method="post">
	<table>
	<tr>
		<td align="right">请输入字符串：</td>
		<td><input type="text" name="sourceStr"   size="30"/></td>
	</tr>
	<tr>
		<td colspan="2" align="center"><input type="submit" value="过  滤" /></td>
	</tr>
	</table>
</form>
```

（3）新建 filterstr.jsp 页，该页用于获取用户输入的表单信息，并通过使用 StringUtil 类来实现字符串过滤，关键代码如下：

```
<%
   String sourceStr = request.getParameter("sourceStr");
%>
<!-- 使用 useBean 动作标签导入 JavaBean 对象 -->
<jsp:useBean id="strBean" class="com.lh.bean.StringUtil"></jsp:useBean>
<!-- 对 StringUtil 类的 sourceStr 属性赋值 -->
<jsp:setProperty property="sourceStr" name="strBean" value="<%=sourceStr %>"/>
<table>
	<tr>
		<td>过滤之前的字符串：</td>
		<td align="left"><jsp:getProperty property="sourceStr" name="strBean"/></td>
	</tr>
	<tr>
		<td>过滤之后的字符串：</td>
		<td align="left"><jsp:getProperty property="targetStr" name="strBean"/></td>
	</tr>
</table>
```

秘笈心法

为了防止向数据库中插入包含大量空格或 NULL 值的冗余数据，应该在数据插入到数据库之前，对数据中

可能包含的空格或 NULL 值进行过滤。

实例 137　**获取汉字的拼音简码**　初级

光盘位置：光盘\MR\05\137　实用指数：★★★☆

实例说明

本实例可以获得用户输入汉字的拼音的第一个字母，这里涉及的是常用汉字，一些特殊汉字没有按照这个规律编码，请根据实际情况另行判断，实例运行效果如图 5.13 所示。



图 5.13　获取汉字的拼音简码

关键技术

本实例首先将输入的字符串转换为数组，然后循环该数组，在循环中使用 String 类的 getBytes()方法将每个字符都转换为字节数组，根据字节数组的长度来判断当前这个字符是汉字还是字母。

如果当前字符为汉字，首先计算出该汉字的 Unicode 编码，然后根据汉字的 Unicode 编码范围来指定当前汉字的首字母。

Unicode 字符集是一种双字节编码方式的字符集，使用 0～65535 之间的双字节无符号整数对每个字符进行编码，可以定义 65536 个不同的字符。Unicode 字符集是对 ASCII 码字符集的扩展，Unicode 字符集中的前 128 个字符对应 ASCII 码字符集中的字符并且与之具有相同的编码值。

设计过程

（1）新建 StringUtil 的 JavaBean 类，该类实现了获取汉字拼音简码的方法，关键代码如下：

```
public class StringUtil {
 private String sourceStr;                              //源字符串
 private String shortPhonetic="";                      //获得汉字的拼音简码字符串
 public String getSourceStr() {
        return sourceStr;
 }
 public void setSourceStr(String sourceStr) {
        this.sourceStr = sourceStr;
 }
 public String getShortPhonetic() {
        char cArr[] = sourceStr.toCharArray();          //将字符串转换为数组
        byte b[] ;
        String sp = "";                                 //该变量用于在循环中保存单个字的拼音首字母
        for(int i=0;i<cArr.length;i++){
               b =String.valueOf(cArr[i]).getBytes();
               if(b.length>1){                          //如果字符串字节长度大于 1，则为汉字
                      int code = 256*(b[0]+256)+(b[1]+256);  //根据字节值获得汉字的 Unicode 无符号编码
                      /**以下代码根据汉字的编码范围指定汉字的拼音首字母*/
                      if (code >= 45217 && code <= 45252) {
                             sp="A";
                      } else if (code >= 45253 && code <= 45760) {
                             sp="B";
                      } else if (code >= 45761 && code <= 46317) {
                             sp="C";
                      } else if (code >= 46318 && code <= 46825) {
                             sp="D";
                      } else if (code >= 46826 && code <= 47009) {
                             sp="E";
                      } else if (code >= 47010 && code <= 47296) {
                             sp="F";
```

```
                    } else if (code >= 47297 && code <= 47613) {
                        sp="G";
                    } else if (code >= 47614 && code <= 48118) {
                        sp="H";
                    } else if (code >= 48119 && code <= 49061) {
                        sp="J";
                    } else if (code >= 49062 && code <= 49323) {
                        sp="K";
                    } else if (code >= 49324 && code <= 49895) {
                        sp="L";
                    } else if (code >= 49896 && code <= 50370) {
                        sp="M";
                    } else if (code >= 50371 && code <= 50613) {
                        sp="N";
                    } else if (code >= 50614 && code <= 50621) {
                        sp="O";
                    } else if (code >= 50622 && code <= 50905) {
                        sp="P";
                    } else if (code >= 50906 && code <= 51386) {
                        sp="Q";
                    } else if (code >= 51387 && code <= 51445) {
                        sp="R";
                    } else if (code >= 51446 && code <= 52217) {
                        sp="S";
                    } else if (code >= 52218 && code <= 52697) {
                        sp="T";
                    } else if (code >= 52698 && code <= 52979) {
                        sp="W";
                    } else if (code >= 52980 && code <= 53640) {
                        sp="X";
                    } else if (code >= 53689 && code <= 54480) {
                        sp="Y";
                    } else if (code >= 54481 && code <= 62289) {
                        sp="Z";
                    }
                }
                else{                                   //如果为英文字母，则直接返回字母本身
                    sp=String.valueOf(cArr[i]);
                }
                shortPhonetic +=sp;                     //将转换后的每个汉字拼音简码连接成一个字符串
            }
            return shortPhonetic;
    }
}
```

（2）新建 index.jsp 页，用于输入表单信息，关键代码如下：

```
<form action="getshortphonetic.jsp" method="post">
    <table>
     <tr>
        <td align="right">请输入汉字：</td>
        <td><input type="text" name="sourceStr"   size="30"/></td>
     </tr>
     <tr>
        <td colspan="2" align="center"><input type="submit" value="获取拼音简码" /></td>
     </tr>
    </table>
</form>
```

（3）新建 getshortphonetic.jsp 页，用于获得表单信息，并通过 StringUtil 类获得汉字字符串的拼音简码，关键代码如下：

```
<%
    String sourceStr = request.getParameter("sourceStr");
%>
<!-- 使用 useBean 动作标签导入 JavaBean 对象 -->
<jsp:useBean id="strBean" class="com.lh.bean.StringUtil"></jsp:useBean>
<!-- 对 StringUtil 类的 sourceStr 属性赋值 -->
<jsp:setProperty property="sourceStr" name="strBean" value="<%=sourceStr %>"/>
<table>
```

```
        <tr>
                <td>输入的汉字：</td>
                <td align="left">
                        <jsp:getProperty property="sourceStr" name="strBean"/>
                </td>
        </tr>
        <tr >
                <td>拼音简码：</td>
                <td align="left">
                        <jsp:getProperty property="shortPhonetic" name="strBean"/>
                </td>
        </tr>
</table>
```

秘笈心法

根据本实例，读者可以设置网上书城中显示图书名称的拼音简码功能，还可以将网页中的信息按中文拼音排序。

5.2　数 据 验 证

在程序开发过程中，为了保持数据的统一完整性和合法性，在数据保存到数据库之前，对数据进行验证这一环节是必不可少的。本节将介绍几个常用的数据验证的 JavaBean 实例。

实例 138	判断字符串是否以指定字符开头 光盘位置：光盘\MR\05\138	初级 实用指数：★★★★

实例说明

在网站注册用户信息时，用户名信息往往不允以数字或者其他特殊字符开头。本实例将介绍如何判断字符串是否以指定字符开头，运行结果如图 5.14 所示。

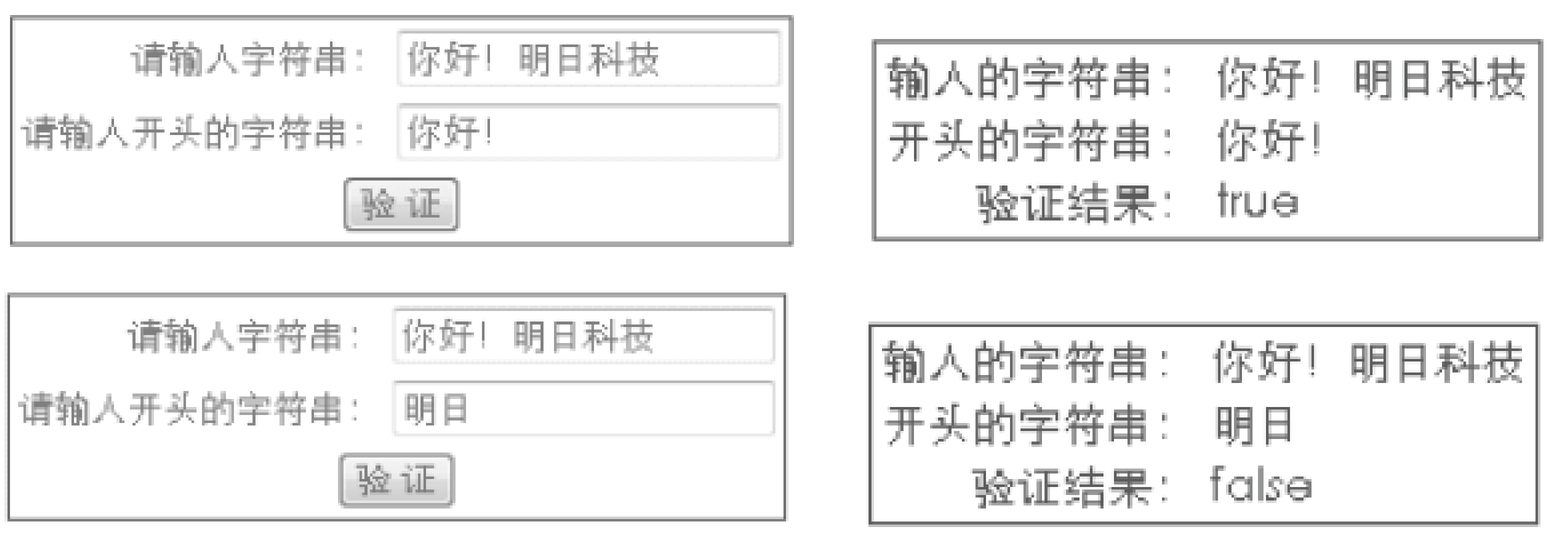


图 5.14　判断字符串是否以指定字符串开头

关键技术

本实例主要应用 String 类中提供的 startsWith()方法来实现，该方法用于判断字符串是否以指定的前缀开始，其语法结构如下：

```
public boolean startsWith(String prefix)
```

参数说明

prefix：为指定的开始字符串，如果字符串以 prefix 开头，则方法返回值为 true，否则返回 false。

在 String 类中还实现了一个同名的重载方法，语法结构如下：

```
public boolean startsWith(String prefix,int toffset)
```

参数说明

❶ prefix：为开始的字符串。

❷ toffset：为子字符串出现的索引位置。

功能：该方法用于测试此字符串从指定索引开始的子字符串是否以指定前缀开始。例如，字符串“abcdefg”，判断第二个索引处是否以“c”开头，代码如下：

```
String str="abcdefg";
str.startsWith("c", 2);
```

设计过程

（1）新建名为 StringUtil 的 JavaBean 类，用于判断字符串是否以指定的子字符串开头，关键代码如下：

```
public class StringUtil {
	private String startStr ;		//指定开头的字符串
	private String str;			//被判断的字符串
	private boolean check;		//判断结果
	public String getStartStr() {
		return startStr;
	}
	public void setStartStr(String startStr) {
		this.startStr = startStr;
	}
	public String getStr() {
		return str;
	}
	public void setStr(String str) {
		this.str = str;
	}
	public boolean isCheck() {
		//使用 startsWith()方法判断字符串是否以指定字符开头，如果是则返回 true，否则返回 false
		return str.startsWith(startStr);
	}
}
```

（2）新建 index.jsp 页，该页用于输入表单信息，关键代码如下：

```
<form action="check.jsp" method="post">
  <table>
      <tr>
        <td align="right">请输入字符串：</td><td><input type="text" name="str"   /></td>
      </tr>
      <tr>
        <td align="right">请输入开头的字符串：</td>
        <td><input type="text" name="startStr"   /></td>
      </tr>
      <tr>
        <td colspan="2" align="center"><input type="submit" value="验  证" /></td>
      </tr>
  </table>
</form>
```

秘笈心法

String 类的 startsWith()方法并不是唯一判断字符串是否以指定字符开头的方法，还可以通过 String 类的 indexOf()方法来实现。indexOf()方法包含 4 种重载方法，这些方法的说明请参见 Java API。

实例 139　检查字符串是否包含英文字母　初级

光盘位置：光盘\MR\05\139　实用指数：★★★★

实例说明

在实际开发过程中，有时需要判断一个字符串是否包含英文字母。本实例将介绍如何检查一个字符串中是否包含英文字母，运行结果如图 5.15 所示。

关键技术

本实例主要应用 String 类的 toCharArray()方法。首先通过该方法将指定的字符串转换为字符数组，然后循

环该字符数组，已知大写英文字母的 ASCII 码范围在 65～90 之间，小写英文字母的 ASCII 码范围在 97～122 之间，所以根据字符的 ASCII 码值就可以判断出字符串中是否包含英文字母。



图 5.15　检查字符串中是否包含英文字母

设计过程

（1）新建名为 StringUtil 的 JavaBean 类，在该类中包含一个判断字符串是否包含英文字母的方法，关键代码如下：

```
public class StringUtil {
	private String str;						//要判断的字符串
	private boolean hasEn;					//是否包含英文字符
	public String getStr() {
		return str;
	}
	public void setStr(String str) {
		this.str = str;
	}
	public boolean isHasEn() {
		char cArr[] = str.toCharArray();		//将字符串转换为字符数组
		boolean b =false;
		StringBuffer sb =new StringBuffer("");
		StringBuffer sb2 =new StringBuffer("");
		for(int i=0;i<cArr.length;i++){
			int ascii = (int)cArr[i];		//强制转换可以直接得到字符的 ASCII 码
			//英文字符的 ASCII 码范围，大写字母 A～Z 的范围是 65～90，小写字母 a～z 的范围是 97～122
			if((ascii>=65&&ascii<=90)||(ascii>=97&&ascii<=122)){
				sb.append(cArr[i]);			//将每个英文字母添加到 StringBuffer 对象中
			}
			else{
				sb2.append(cArr[i]);
			}
		}
		if(!sb.toString().equals(""))			//如果保存英文字母的字符串不为""，说明该字符串包含英文字母
			hasEn=true;
		else
			hasEn=false;
		return hasEn;
	}
	public void setHasEn(boolean hasEn) {
		this.hasEn = hasEn;
	}
}
```

（2）新建 index.jsp 页，用于输入表单信息，关键代码如下：

```
<form action="check.jsp" method="post">
	<table>
	<tr>
		<td align="right">请输入字符串：</td>
		<td><input type="text" name="str"   size="30"/></td>
	</tr>
	<tr>
		<td colspan="2" align="center"><input type="submit" value="检　查" /></td>
	</tr>
	</table>
</form>
```

（3）新建 check.jsp 页，用于获得表单信息，并通过 StringUtil 类中的方法判断字符串是否包含英文字母，关键代码如下：

```
<%
  String str = request.getParameter("str");
%>
<!-- 使用 useBean 动作标签导入 JavaBean 对象 -->
```

```
<jsp:useBean id="strBean" class="com.lh.bean.StringUtil"></jsp:useBean>
<!-- 对 StringUtil 类的 str 属性赋值 -->
<jsp:setProperty property="str" name="strBean" value="<%=str %>"/>
<table>
        <tr>
                <td align="right">输入的字符串：</td>
                <td>
                        <!-- 从 StringUtil 对象中获得 str 的属性值 -->
                        <jsp:getProperty property="str" name="strBean"/>
                </td>
        </tr>
        <tr><td align="right">是否包含英文字母：</td>
                <td>
                        <!-- 从 StringUtil 对象中获得 hasEn 的属性值 -->
                        <jsp:getProperty property="hasEn" name="strBean"/>
                </td>
        </tr>
        <tr>
                <td align="right">去掉英文字母后的字符串：</td>
                <td>
                        <!-- 从 StringUtil 对象中获得 cnStr 的属性值 -->
                        <jsp:getProperty property="cnStr" name="strBean"/>
                </td>
        </tr>
</table>
```

秘笈心法

通过 String 类的 hashCode()方法也可以实现本实例，在类的 main()方法中编写如下代码：

```
String str = "abcd";
char cArr[] = str.toCharArray();
for(int i=0;i<cArr.length;i++){
        int code = String.valueOf(cArr[i]).hashCode();
        System.out.println(code);
}
```

以上代码执行后，会分别输出 a、b、c、d 的 ASCII 码值。

实例 140 检查字符串是否包含数字

光盘位置：光盘\MR\05\140

初级

实用指数：★★★★

实例说明

在实际开发过程中，有时需要判断一个字符串是否包含数字。本实例将介绍如何判断字符串是否包含数字，其运行效果如图 5.16 所示。

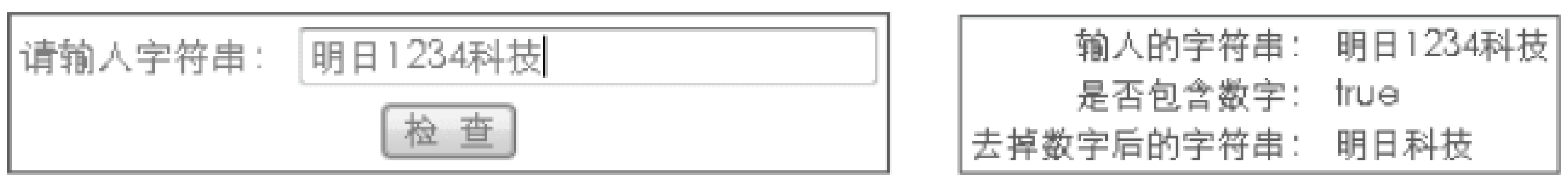


图 5.16　判断字符串是否包含数字

关键技术

本实例的实现过程与实例 139 的实现过程大同小异，都是通过循环字符数组，根据字符的 ASCII 码值来判断字符是否为字母或数字。数字 0～9 的 ASCII 码值范围在 48～57 之间，所以在循环中，根据判断每个字符的 ASCII 码是否在 48～57 之间，就可以知道字符串是否包含数字。

设计过程

新建 StringUtil 的 JavaBean 类，该类用于判断字符串是否包含数字，关键代码如下：

```
public class StringUtil {
 private String   str;                                    //要检查的字符串
```

```
private boolean hasNum;                    //是否包含数字
private String othersStr;                  //去掉数字后的字符串
public boolean isHasNum() {
    char cArr[] = str.toCharArray();       //将字符串转换为字符数组
    StringBuffer sbNum =new StringBuffer("");
    StringBuffer sbOthers =new StringBuffer("");
    for(int i=0;i<cArr.length;i++){
        int ascii = (int)cArr[i];          //强制转换可以直接得到字符的 ASCII 码
        if(ascii>=48&&ascii<=57){          //数字的 ASCII 码范围在 48～57 之间
            sbNum.append(cArr[i]);         //将每个数字添加到 StringBuffer 对象中
        }
        else{
            sbOthers.append(cArr[i]);
        }
    }
    this.setOthersStr(sbOthers.toString());
    if(!sbNum.toString().equals(""))
        hasNum=true;
    else
        hasNum=false;
    return hasNum;
}
            …//此处省略了属性的 get 和 set 方法
}
```

秘笈心法

本实例还可以通过 String 类的 hashCode()方法实现。

实例 141　判断用户输入的日期是否为当前日期　初级

光盘位置：光盘\MR\05\141　实用指数：★★★★

实例说明

在实际开发过程中，会遇到判断用户输入的日期是否为当前日期的情况。本实例将介绍如何判断用户输入的日期是否为系统当前日期，运行效果如图 5.17 所示。

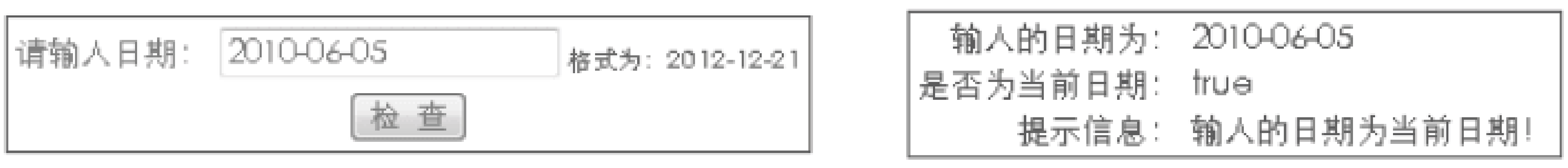


图 5.17　判断用户输入的日期是否为当前日期

关键技术

本实例主要应用 java.util.Calendar 类实现，首先使用该类的 getInstance()方法来获取系统当前时间的日历对象，然后使用该对象中提供的一系列方法可以获得当前时间的年、月、日、小时、分、秒等值。

- 创建一个当前时间的 Calendar 对象，代码如下：

```
Calendar now = Calendar.getInstance();
```

- 获得当前时间的年份，代码如下：

```
int year = now.get(now.YEAR);
```

- 获得当前时间的月份，代码如下：

```
int month = now.get(now.MONTH)+1;//默认的月份是从 0 开始的，所以需要加 1
```

- 获得当前时间的日，代码如下：

```
int date = now.get(now.DAY_OF_MONTH);
```

- 获得当前时间的小时，代码如下：

```
int hour = now.get(now.HOUR_OF_DAY);
```

- 获得当前时间的分钟，代码如下：

```
int minute = now.get(now.MINUTE);
```

❑ 获得当前时间的秒，代码如下：

```
int second = now.get(now.SECOND);
```

设计过程

（1）新建名为 StringUtil 的 JavaBean 类，该类用于判断用户输入的日期是否为当前日期，关键代码如下：

```
public class StringUtil {
 private String dateStr;                                    //用户输入的日期
 private boolean today;                                     //判断是否为今天
 private String cue;                                        //提示信息
 public StringUtil(){}
 public String getDateStr() {
        return dateStr;
 }
 public void setDateStr(String dateStr) {
        dateStr = dateStr.replaceAll(" ","");               //替换日期中的空格为""
        this.dateStr = dateStr;
 }
 public boolean isToday() {
        String dateArr[] = dateStr.split("-");              //将日期字符串分解为数组
        int year =Integer.parseInt(dateArr[0]);
        int month =Integer.parseInt(dateArr[1]);
        int date = Integer.parseInt(dateArr[2]);
        Calendar now = Calendar.getInstance();              //获得系统当前时间的 Calendar 对象
        int nowYear = now.get(now.YEAR);                    //获得当前时间的年
        int nowMonth = now.get(now.MONTH)+1;                //获得当前时间的月
        int nowDate = now.get(now.DAY_OF_MONTH);            //获得当前时间的日
        if(year==nowYear&&month==nowMonth&&date==nowDate){ //如果年月日的值都相同
               cue="输入的日期为当前日期！";
               today=true;
        }else{
               cue="输入的日期不是当前日期！";
               today=false;
        }
        return today;
 }
……                                                          //省略了一些属性的 get 和 set 方法
}
```

（2）新建 index.jsp 页，用于输入表单信息，关键代码如下：

```
<form action="check.jsp" method="post">
       <table>
        <tr>
               <td align="right">请输入日期：</td>
               <td><input type="text" name="datestr" /><font>格式为：2012-12-21</font></td>
        </tr>
        <tr>
               <td colspan="2" align="center"><input type="submit" value="检    查" /></td>
        </tr>
       </table>
</form>
```

（3）新建 check.jsp 页，判断用户输入的日期是否为当前日期，关键代码如下：

```
<%
    String dateStr = request.getParameter("datestr");
%>
<!-- 使用 useBean 动作标签导入 JavaBean 对象 -->
<jsp:useBean id="strBean" class="com.lh.bean.StringUtil"></jsp:useBean>
<!-- 对 StringUtil 类的 dateStr 属性赋值 -->
<jsp:setProperty property="dateStr" name="strBean" value="<%=dateStr %>"/>
<table>
        <tr>
               <td align="right">输入的日期为：</td>
               <td>
                      <!-- 从 StringUtil 对象中获得 dateStr 的属性值 -->
                      <jsp:getProperty property="dateStr" name="strBean"/>
               </td>
```

```
        </tr>
        <tr>
            <td align="right">是否为当前日期：</td>
            <td>
                <!-- 从 StringUtil 对象中获得 today 的属性值 -->
                <jsp:getProperty property="today" name="strBean"/>
            </td>
        </tr>
        <tr><td align="right">提示信息：</td>
            <td>
                <!-- 从 StringUtil 对象中获得 cue 的属性值 -->
                <jsp:getProperty property="cue" name="strBean"/>
            </td>
        </tr>
</table>
```

秘笈心法

本实例也可以通过 java.util.Date 类实现，Date 类表示特定的瞬间，精确到毫秒。通过 Date 类的 getXXX()方法可以获得 Date 对象中的年、月、日、小时、分、秒，通过 setXXX()方法可以设置 Date 对象中的年、月、日、小时、分、秒。但是 Date 类的这些方法从 JDK 1.1 开始，已经被 Calendar 类中相应的方法取代。所以在实际的开发中，应该尽量使用 Calendar 类。

实例 142　判断是否为数字

光盘位置：光盘\MR\05\142

初级

实用指数：★★★★

实例说明

在网站中填写表单信息时，有些字段信息必须是数字格式的，如用户的年龄、工资收入、手机号码等。本实例将介绍如何判断用户输入的数据是否为数字，其运行效果如图 5.18 所示。

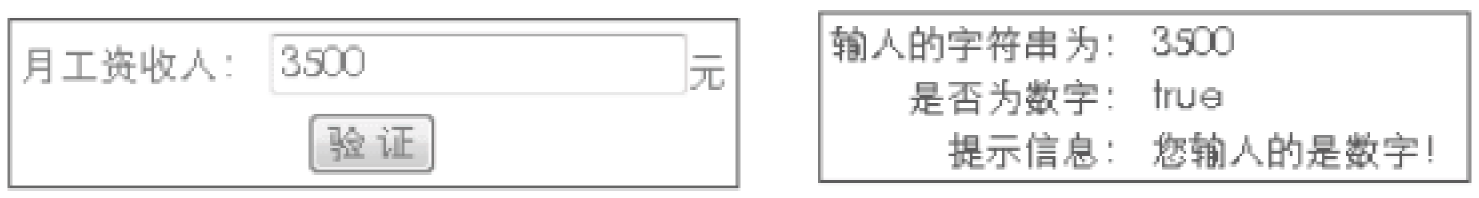


图 5.18　判断是否为数字

关键技术

本实例主要通过字符的 ASCII 码值来判断数据是否为数字，数字的 ASCII 码值的范围在 48～57 之间。首先将整个字符串通过 String 类的 toCharArray()方法转换为字符数组，然后在循环中把每个字符转换为 ASCII 码，判断是否在 4～57 之间，如果条件满足则使用 StringBuffer 对象累加每个字符，最后判断 StringBuffer 对象的字符串长度是否等于字符数组的长度，如果相等则说明整个字符串为数字。

设计过程

（1）新建名为 StringUtil 的 JavaBean 类，该类主要用于验证字符串是否为数字，关键代码如下：

```
public class StringUtil {
 private String numStr;                          //要判断的字符串
 private boolean number;                         //判断结果
 private String cue ;                            //提示信息
 public void setNumStr(String numStr) {
        numStr= numStr.replaceAll(" ", "");      //去掉字符串中的空格
        this.numStr = numStr;
 }
 public boolean isNumber() {
        char cArr[]= numStr.toCharArray();       //将字符串转换为字符数组
        StringBuffer sb = new StringBuffer("");  //创建动态字符串对象
        for(int i=0;i<cArr.length;i++){
```

```
            int ascii = (int)cArr[i];           //将字符强制转换为 int 值，该值为字符的 ASCII 码
            if(ascii>=48&&ascii<=57){           //数字 0～9 的 ASCII 码范围在 48～57 之间
                sb.append(cArr[i]);             //条件满足，将字符添加到 StringBuffer 字符串末尾
            }
        }
        if(sb.length()==cArr.length){           //如果 StringBuffer 字符串的长度等于字符数组的长度
            number=true;                        //该字符串为数字
            this.setCue("您输入的是数字！");
        }
        else{
            number=false;
            this.setCue("您输入的不是数字！");
        }
        return number;
}
……          //此处省略了部分 get 和 set 方法
}
```

（2）新建 index.jsp 页，用于输入表单信息，关键代码如下：

```
<form action="check.jsp" method="post">
    <table>
        <tr>
            <td align="right">月工资收入：</td>
            <td><input type="text" name="numstr" />元</td>
        </tr>
        <tr>
            <td colspan="2" align="center"><input type="submit" value="验  证" /></td>
        </tr>
    </table>
</form>
```

（3）新建 check.jsp 页，验证用户输入的字符串是否为数字，关键代码如下：

```
<%
    String numStr = request.getParameter("numstr");
%>
<!-- 使用 useBean 动作标签导入 JavaBean 对象 -->
<jsp:useBean id="strBean" class="com.lh.bean.StringUtil"></jsp:useBean>
<!-- 对 StringUtil 类的 numStr 属性赋值 -->
<jsp:setProperty property="numStr" name="strBean" value="<%=numStr %>"/>
<table>
    <tr>
        <td align="right">输入的字符串为：</td>
        <td>
            <!-- 从 StringUtil 对象中获得 numStr 的属性值 -->
            <jsp:getProperty property="numStr" name="strBean"/>
        </td>
    </tr>
    <tr>
        <td align="right">是否为数字：</td>
        <td>
            <!-- 从 StringUtil 对象中获得 numStr 的属性值 -->
            <jsp:getProperty property="number" name="strBean"/>
        </td>
    </tr>
    <tr>
        <td align="right">提示信息：</td>
        <td>
            <!-- 从 StringUtil 对象中获得 cue 的属性值 -->
            <jsp:getProperty property="cue" name="strBean"/>
        </td>
    </tr>
</table>
```

■ 秘笈心法

本实例主要是将一个字符串分解为字符数组，然后判断每个字符是否为数字，如果是数字就添加到 StringBuffer 对象中，最后把这个 StringBuffer 对象的长度与输入字符串的长度对比，判断输入的整个字符串是否为数字。

实例 143　判断用户名是否有效

光盘位置：光盘\MR\05\143

初级

实用指数：★★★★

实例说明

有些网站用户注册时，用户名信息只能由字母、数字和下划线组成，并且首字符必须为字母，用户名中不允许包含特殊字符等。本实例将介绍如何判断一个用户名是否有效，其运行效果如图 5.19 所示。

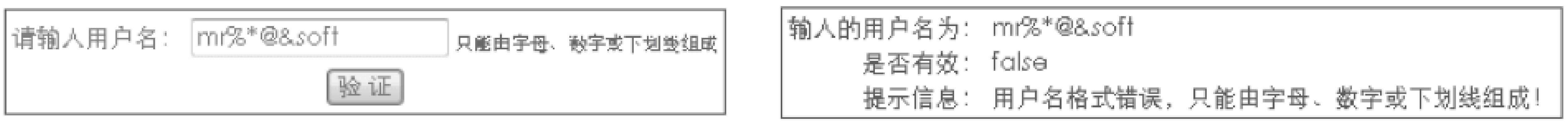


图 5.19　判断用户名是否有效

关键技术

本实例主要应用字符的 ASCII 码来判断用户名是否有效，其中英文字母的 ASCII 范围是 65～90（大写字母）和 97～122（小写字母），下划线的 ASCII 码为 95，数字的 ASCII 码范围在 47～58 之间。

设计过程

（1）新建名为 StringBuffer 的 JavaBean 类，该类用于判断用户名是否有效，关键代码如下：

```
public class StringUtil {
 private String str;                          //要判断的字符串
 private boolean valid;                       //是否有效
 private String cue;                          //提示信息
 public boolean isValid() {                   //boolean 属性的 get 方法写法为 isXXX()
        char cArr[] =str.toCharArray();       //字符串转换为字符数组
        int firstChar =(int) cArr[0];         //第一个字符的 ASCII 码
        StringBuffer sb = new StringBuffer("");
        if((firstChar>=65&&firstChar<=90)||(firstChar>=97&&firstChar<=122)){ //判断首字符是否为字母
               for(int i=1;i<cArr.length;i++){
                      int ascii =cArr[i];     //获得字符的 ASCII 码
                      //判断字符是否为字母、数字或下划线，下划线的 ASCII 码为 95
                      if((ascii>=48&&ascii<=57)||(ascii>=65&&ascii<=90)|| (ascii==95)||(ascii>=97&&ascii<=122)){
                             sb.append(cArr[i]);   //如果条件满足，将字符添加到 StringBuffer 字符串的末尾
                      }
               }
               int length = cArr.length-sb.toString().length();
               if(length==1){ //如果被判断字符串的长度与 StringBuffer 字符串记录的长度差 1（即去掉首字符的长度）
                      this.setCue("用户名格式正确！");
                      valid=true;
                      return valid;
               }else{
                      this.setCue("用户名格式错误，只能由字母、数字或下划线组成！");
                      valid=false;
                      return valid;
               }
        }else{                                //如果首字符不是字母，直接返回 false
               this.setCue("用户名格式不对，首字符必须为字母！");
               valid=false;
               return valid;
        }
}
… //此处省略了其他属性的 get 和 set 方法
}
```

（2）新建 index.jsp 页，用于输入表单信息，关键代码如下：

```
<form action="check.jsp" method="post">
       <table>
        <tr>
```

```
                <td align="right">请输入用户名：</td>
                <td><input type="text" name="name" />
                        <font>只能由字母、数字或下划线组成</font></td>
        </tr>
        <tr>
                <td colspan="2" align="center"><input type="submit" value="验  证" /></td>
        </tr>
      </table>
</form>
```

（3）新建 check.jsp 页，用于判断用户名是否有效，关键代码如下：

```
<%
        String name = request.getParameter("name");
%>
<!-- 使用 useBean 动作标签导入 JavaBean 对象 -->
<jsp:useBean id="strBean" class="com.lh.bean.StringUtil"></jsp:useBean>
<!-- 对 StringUtil 类的 str 属性赋值 -->
<jsp:setProperty property="str" name="strBean" value="<%=name %>"/>
<table>
        <tr>
                <td align="right">输入的用户名为：</td>
                <td>
                        <!-- 从 StringUtil 对象中获得 Str 的属性值 -->
                        <jsp:getProperty property="str" name="strBean"/>
                </td>
        </tr>
        <tr>
                <td align="right">是否有效：</td>
                <td>
                        <!-- 从 StringUtil 对象中获得 valid 的属性值 -->
                        <jsp:getProperty property="valid" name="strBean"/>
                </td>
        </tr>
        <tr>
                <td align="right">提示信息：</td>
                <td>
                        <!-- 从 StringUtil 对象中获得 cue 的属性值 -->
                        <jsp:getProperty property="cue" name="strBean"/>
                </td>
        </tr>
</table>
```

秘笈心法

在本实例中，将字符转换为 ASCII 码时，还可以通过 String 类的 hashCode()方法实现。

5.3 日期时间处理

在程序的开发过程中，经常需要处理日期时间类型的数据，如用户在表单中输入的为字符串类型的日期，而数据库中保存的是日期时间类型的数据，所以在保存数据库之前需要将字符串转换为日期时间类型。有时页面中显示的数据为简写的日期字符串，所以需要将一个日期时间类型的对象转换为简写的字符串。本节将介绍几个常用的对日期时间操作的 JavaBean 实例。

实例 144	将指定日期字符串转换为 Calendar 对象 光盘位置：光盘\MR\05\144	初级 实用指数：★★★★

实例说明

在实际开发过程中，有时需要将用户输入的日期字符串转换为 Calendar 类型。本实例介绍的是如何将一个

日期字符串转换为 Calendar 对象类型，其运行效果如图 5.20 所示。

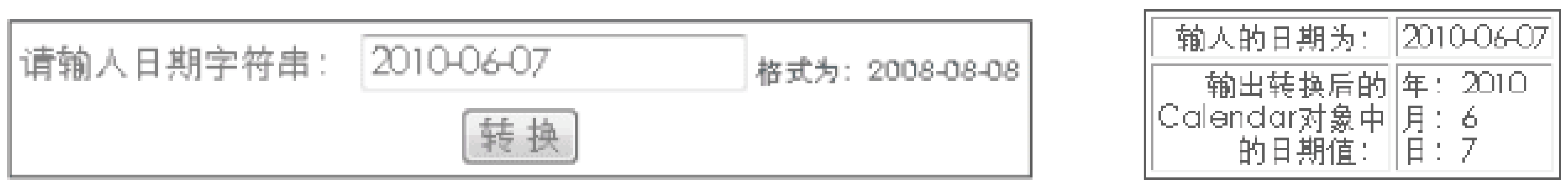


图 5.20　将日期字符串转换为 Calendar 对象

关键技术

本实例主要应用格式化日期时间的 java.text.SimpleDateFormat 类。已知日期字符串的格式为“yyyy-mm-dd”，如“2010-06-07”，实现步骤如下：

（1）创建一个“yyyy-mm-dd”格式的格式化对象，具体代码如下：

```
SimpleDateFormat format = new SimpleDateFormat("yyyy-mm-dd");
```

（2）通过 SimpleDateFormat 对象的 parse()方法将指定字符串转换为 Date 对象，具体代码如下：

```
Date date = format.parse("2010-06-07");
```

（3）通过 Calendar 对象的 setTime()方法将 Date 对象转换为 Calendar 对象，具体代码如下：

```
Calendar calendar = Calendar.getInstance();
calendar.setTime(date);
```

设计过程

（1）新建名为 StringUtil 的 JavaBean 类，用于将日期字符串转换为 Calendar 对象，关键代码如下：

```
public class StringUtil {
 private String dateStr;                                    //日期字符串
 private Calendar calendar=null;                            //将字符串转换后的 Calendar 对象
 public Calendar getCalendar() {
        Date date =null;                                    //声明一个 Date 类型的对象
        SimpleDateFormat format = null;                     //声明格式化日期的对象
        if(dateStr!=null){
               format = new SimpleDateFormat("yyyy-MM-dd");   //创建日期的格式化类型
               calendar = Calendar.getInstance();           //创建一个 Calendar 类型的对象
               try {                                        //forma.parse()方法会抛出异常
                      date = format.parse(dateStr);         //解析日期字符串，生成 Date 对象
                      calendar.setTime(date);               //使用 Date 对象设置此 Calendar 对象的时间
               } catch (ParseException e) {
                      e.printStackTrace();
               }
        }
        return calendar;
}
……            //省略了其他属性的 get 和 set 方法
}
```

（2）新建 index.jsp 页，用于输入表单信息，关键代码如下：

```
<form action="tocalendar.jsp" method="post">
      <table>
       <tr>
              <td align="right">请输入日期字符串：</td>
              <td><input type="text" name="datestr" />
                     <font>格式为：2008-08-80</font>
              </td>
       </tr>
       <tr>
              <td colspan="2" align="center"><input type="submit" value="转  换" /></td>
       </tr>
      </table>
</form>
```

（3）新建 tocalendar.jsp 页，用于将日期字符串转换为 Calendar 对象，并输入 Calendar 对象中的年、月、日，关键代码如下：

```
<%
       String dateStr = request.getParameter("datestr");
%>
<!-- 使用 useBean 动作标签导入 JavaBean 对象  -->
```

```
<jsp:useBean id="strBean" class="com.lh.bean.StringUtil"></jsp:useBean>
<!-- 对 StringUtil 类的 dateStr 属性赋值 -->
<jsp:setProperty property="dateStr" name="strBean" value="<%=dateStr %>"/>
<table border="1">
	<tr>
		<td align="right">输入的日期为：</td>
		<td>
			<!-- 从 StringUtil 对象中获得 dateStr 的属性值 -->
			<jsp:getProperty property="dateStr" name="strBean"/>
		</td>
	</tr>
	<tr>
		<td align="right" width="100">输出转换后的 Calendar 对象中的日期值：</td>
		<td >
			<%
				Calendar calendar = strBean.getCalendar();
			 %>
			 年：<%=calendar.get(calendar.YEAR)%><br>
			 月：<%=calendar.get(calendar.MONTH)+1%><br>
			 日：<%=calendar.get(calendar.DAY_OF_MONTH)%>
		</td>
	</tr>
</table>
```

秘笈心法

在实际的应用开发过程中，经常需要将日期字符串转换为 Calendar 对象类型。因此读者有必要掌握如何将一个日期字符串转换为 Calendar 对象。

实例 145 将 Calendar 对象转换为日期时间字符串

光盘位置：光盘\MR\05\145

初级

实用指数：★★★★

实例说明

在网页中显示数据时，需要将 Calendar 对象转换为日期时间字符串再进行显示。本实例将讲解如何将一个 Calendar 类型的对象转换为指定格式的日期时间字符串，其运行结果如图 5.21 所示。



图 5.21　将 Calendar 对象转换为日期时间字符串

关键技术

本实例主要应用 Calendar 类的 get()方法，使用该方法可以获得 Calendar 对象中的年、月、日、小时、分钟、秒等。

设计过程

（1）新建名为 StringUtil 的 JavaBean 类，用于将 Calendar 对象转换为指定格式的日期字符串，关键代码如下：

```
public class StringUtil {
	private String dateStr;                                        //日期字符串
	private Calendar calendar=null;                                //将字符串转换后的 Calendar 对象
	public String getDateStr() {
		int year = calendar.get(calendar.YEAR);                    //获得年
		int month = calendar.get(calendar.MONTH)+1;                //获得月
		int date = calendar.get(calendar.DAY_OF_MONTH);            //获得日
		dateStr = year+"-"+month+"-"+date;                         //生成日期字符串，格式为 2008-8-8
		return dateStr;
```

```
}
……          //省略了其他属性的 get 和 set 方法
}
```

（2）新建 index.jsp 页，用于输入表单信息，关键代码如下：

```
<form action="tocalendar.jsp" method="post">
    <table>
      <tr>
        <td align="right">请输入日期字符串：</td>
        <td><input type="text" name="datestr" />
            <font>格式为：2008-08-08</font>
        </td>
      </tr>
      <tr>
        <td colspan="2" align="center"><input type="submit" value="转  换" /></td>
      </tr>
    </table>
</form>
```

（3）新建 todatestr.jsp 页，该页首先获得表单的日期字符串，将字符串中的日期转换为一个 Calendar 对象，然后再调用 StringUtil 类将这个 Calendar 对象转换为日期字符串，关键代码如下：

```
<%
    String dateStr = request.getParameter("datestr");        //获得表单中的字符串
    String date[] =dateStr.split("-");                       //根据"-"分隔字符串为数组
    int y =Integer.parseInt(date[0]);                        //获得字符串中的年
    int m =Integer.parseInt(date[1]);                        //获得字符串中的月
    int d = Integer.parseInt(date[2]);                       //获得字符串中的日
    Calendar c = Calendar.getInstance();                     //创建一个 Calendar 对象
    c.set(y,m-1,d);                                          //设置 Calendar 对象的年月日
%>
<!-- 使用 useBean 动作标签导入 JavaBean 对象 -->
<jsp:useBean id="strBean" class="com.lh.bean.StringUtil"></jsp:useBean>
<!-- 对 StringUtil 类的 calendar 属性赋值 -->
<jsp:setProperty property="calendar" name="strBean" value="<%=c %>"/>
<table>
    <tr>
        <td align="right">转换为日期字符串：</td>
        <td>
            <!-- 从 StringUtil 对象中获得 dateStr 的属性值 -->
            <jsp:getProperty property="dateStr" name="strBean"/>
        </td>
    </tr>
</table>
```

秘笈心法

在实际开发应用中，有时需要将 Calendar 对象转换为不同格式的日期时间字符串。例如，格式可能为“yyyy-mm-dd”“yyyy-mm-dd hh:mm:ss”“yyyy/mm/dd”“yyyy 年 mm 月 dd 日”。所以，读者应该掌握本实例的实现方法。

实例 146　获得系统当前时间的字符串格式

初级

光盘位置：光盘\MR\05\146

实用指数：★★★★

实例说明

在实际开发过程中，有时需要获取系统当前的时间，如用户注册时的注册时间，虽然用户不需要填写注册时间，但是在后台需要获取当前的系统时间作为注册时间。本实例介绍的是如何获得系统当前时间的字符串格式，其运行结果如图 5.22 所示。



图 5.22　获得系统当前时间的字符串格式

关键技术

本实例主要通过 java.util.Calendar 类以及 java.util.Date 类来获得系统的当前时间。

使用 Calendar 类时，要通过该类对象的 get()方法获得时间中的年、月、日、小时、分钟和秒，然后将这些值组成一个字符串。

使用 Date 类时，可以通过 java.text.SimpleDateFormat 类将一个 Date 对象格式化为指定格式的日期时间字符串，代码如下：

```
Date date = new Date();
SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd hh-mm-ss");
String dateStr = format.format(date);
```

设计过程

（1）新建名为 StringUtil 的 JavaBean 类，该类实现和获得系统当前时间的两种方法，关键代码如下：

```
public class StringUtil {
 private String timeStr1;                              //日期字符串
 private String timeStr2;
 public String getTimeStr1() {                         //使用 Calendar 对象获得系统当前时间的方法
        Calendar c = Calendar.getInstance();           //创建表示当前时间的 Calendar 对象
        int year = c.get(c.YEAR);                      //获得当前时间的年
        int month =c.get(c.MONTH)+1;                   //获得当前时间的月
        int date = c.get(c.DAY_OF_MONTH);              //获得当前时间的日
        int hour = c.get(c.HOUR_OF_DAY);               //获得当前时间的小时
        int minute = c.get(c.MINUTE);                  //获得当前时间的分钟
        int second = c.get(c.SECOND);                  //获得当前时间的秒
        timeStr1 = year+"-"+month+"-"+date+" "+hour+":"+minute+":"+second;
        return timeStr1;
 }
 public String getTimeStr2() {                         //使用 Date 对象获得系统当前时间的方法
        Date date = new Date();                        //创建表示当前时间的 Date 对象
        //创建格式化日期时间对象
        SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
        timeStr2 = format.format(date);                //格式化当前时间的 Date 对象
        return timeStr2;
 }
}
```

（2）新建 index.jsp 页，在该页中通过 StringUtil 类来获得系统的当前时间字符串，关键代码如下：

```
<!-- 使用 useBean 动作标签导入 JavaBean 对象 -->
<jsp:useBean id="strBean" class="com.lh.bean.StringUtil"></jsp:useBean>
    <table>
      <tr>
            <td align="right">使用 Calendar 对象获得的当前时间：</td>
            <td>
                  <!-- 从 StringUtil 对象中获得 timeStr1 的属性值 -->
                  <jsp:getProperty property="timeStr1" name="strBean"/>
            </td>
      </tr>
      <tr>
            <td align="right" >使用 Date 对象获得的当前时间：</td>
            <td >
                  <!-- 从 StringUtil 对象中获得 timeStr2 的属性值 -->
                  <jsp:getProperty property="timeStr2" name="strBean"/>
            </td>
      </tr>
</table>
```

秘笈心法

在实际的开发应用中，经常需要根据当前系统时间来实现一些业务逻辑，因此读者应该掌握如何获取当前的系统时间。

实例 147 计算出两个日期相差的天数

光盘位置：光盘\MR\05\147

初级

实用指数：★★★★

实例说明

在实际开发过程中，有时会遇到计算两个日期相差天数的情况。本实例将讲解如何计算出两个日期相差的天数，其运行效果如图 5.23 所示。

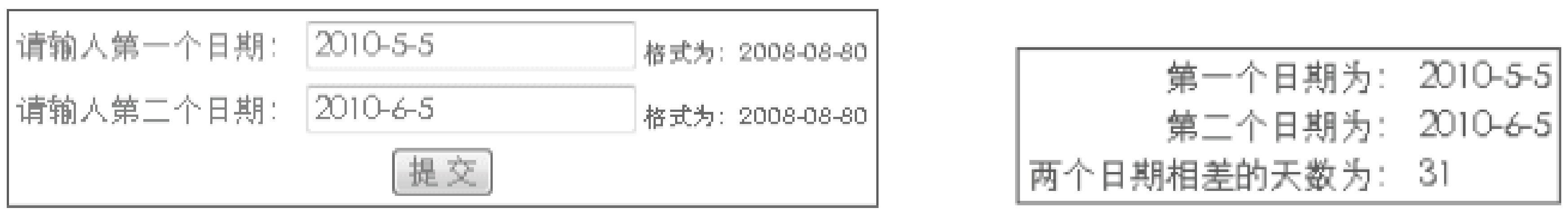


图 5.23 计算出两个日期相差的天数

关键技术

本实例主要应用 Calendar 对象中的 getTimeInMillis()方法，此方法返回一个 long 型的时间值，以毫秒为单位。首先获得两个日期对象的 long 类型的时间值，然后这两个值相减之后再除以一天的毫秒值（1000*60*60*24），得到的值取整即两个日期相差的天数。

设计过程

（1）新建名为 StringUtil 的 JavaBean 类，该类用于计算两个日期相差的天数，关键代码如下：

```
public class StringUtil {
 private String dateStr1;                          //第一个日期字符串
 private String dateStr2;                          //第二个日期字符串
 private int minus;                                //两个日期的差
 public int getMinus() {
        Calendar c1 = this.getCalendar(dateStr1);  //根据第一个日期字符串获得 Calendar 对象
        Calendar c2 = this.getCalendar(dateStr2);  //根据第二个日期字符串获得 Calendar 对象
        long t1 = c1.getTimeInMillis();            //获得此对象的时间值，以毫秒为单位
        long t2 = c2.getTimeInMillis();            //获得此对象的时间值，以毫秒为单位
        long t = 1000*60*60*24;                    //一天的毫秒：1000 毫秒*60 秒*60 分钟*24 小时
        minus =   (int)((t2-t1)/t);                //获得两个日期相差的天数
        return minus;
 }
 public Calendar getCalendar(String dateStr) {
        Date date =null;                           //声明一个 Date 类型的对象
        SimpleDateFormat format = null;            //声明格式化日期的对象
        Calendar calendar = null;
        if(dateStr!=null){
                format = new SimpleDateFormat("yyyy-MM-dd");//创建日期的格式化类型
                calendar = Calendar.getInstance();   //创建一个 Calendar 类型的对象
            try {                                    //forma.parse()方法会抛出异常
                date = format.parse(dateStr);        //解析日期字符串，生成 Date 对象
                calendar.setTime(date);              //使用 Date 对象设置此 Calendar 对象的时间
            } catch (ParseException e) {
                e.printStackTrace();
            }
        }
        return calendar;
 }
……          //此处省略了其他属性的 get 和 set 方法
}
```

（2）新建 index.jsp 页，用于输入表单信息，关键代码如下：

```
<form action="getminus.jsp" method="post">
    <table>
      <tr>
           <td align="right">请输入第一个日期：</td>
```

```
            <td><input type="text" name="datestr1" /><font>格式为：2008-08-80</font></td>
        </tr>
        <tr>
            <td align="right">请输入第二个日期：</td>
            <td><input type="text" name="datestr2" /><font>格式为：2008-08-80</font></td>
        </tr>
        <tr>
            <td colspan="2" align="center"><input type="submit" value="提 交" /></td>
        </tr>
    </table>
</form>
```

（3）新建 getminus.jsp 页，在该页中通过 StringUtil 提供的方法计算出两个日期相差的天数，关键代码如下：

```
<%
        String dateStr1 = request.getParameter("datestr1");
        String dateStr2 = request.getParameter("datestr2");
%>
<!-- 使用 useBean 动作标签导入 JavaBean 对象 -->
<jsp:useBean id="strBean" class="com.lh.bean.StringUtil"></jsp:useBean>
<!-- 对 StringUtil 类的属性赋值 -->
<jsp:setProperty property="dateStr1" name="strBean" value="<%=dateStr1 %>"/>
<jsp:setProperty property="dateStr2" name="strBean" value="<%=dateStr2 %>"/>
<table>
        <tr>
            <td align="right">第一个日期为：</td>
            <td>
                <!-- 从 StringUtil 对象中获得属性值 -->
                <jsp:getProperty property="dateStr1" name="strBean"/>
            </td>
        </tr>
        <tr>
            <td align="right">第二个日期为：</td>
            <td>
                <jsp:getProperty property="dateStr2" name="strBean"/></td>
        </tr>
        <tr >
            <td align="right">两个日期相差的天数为：</td>
            <td >
                <jsp:getProperty property="minus" name="strBean"/>
            </td>
        </tr>
</table>
```

秘笈心法

在实际开发过程中，经常需要计算两个日期相差的天数，所以读者应该掌握本实例的实现方法。

5.4 输出实用的 HTML 代码

在网站开发的过程中，经常会在多个页面使用相同的 HTML 代码。例如，在浏览信息的模块中编写的分页导航链接的代码；在页尾输出的网站版权信息的代码等。本节将介绍几个常用的输出实用的 HTML 代码的 JavaBean 实例。

实例 148	输出提示信息的方法 光盘位置：光盘\MR\05\148	初级 实用指数：★★★★

实例说明

运行本实例，在 index.jsp 页中输入用户名和密码并将其提交到 check.jsp 页，在 check.jsp 页中会对用户名和密码进行判断，如果用户名和密码正确，则输出登录成功信息，否则输出登录失败信息，运行结果如图 5.24 所示。



图 5.24　验证用户登录并输出提示信息

关键技术

本实例主要在 JavaBean 中获得用户输入的用户名和密码，然后根据用户名和密码信息来生成不同的 HTML 标签内容。

设计过程

（1）新建名为 StringUtil 的 JavaBean 类，用于判断用户名和密码是否正确并生成提示信息，关键代码如下：

```
public class StringUtil {
 private String name;        //用户名
 private String pwd ;        //密码
 private String cue;         //HTML 提示信息
 public String getCue() {
        //如果用户名或密码为 null 或""
        if((name==null||name.equals(""))||(pwd==null||pwd.equals(""))){
                cue= "请输入用户名或密码！ ";
        }else{//如果用户名和密码不为空
                if(name.equals("mr")&&pwd.equals("123")){
                        cue= "登录成功！ ";
                }else{
                        cue= "登录失败！  <b>用户名</b>或<b>密码</b>不正确！ ";
                }
        }
        return cue;
 }
……              //省略了其他属性的 get 和 set 方法
}
```

（2）新建 index.jsp 页，用于输入表单信息，关键代码如下：

```
<form action="login.jsp" method="post">
       <table>
        <tr>
                <td align="right">请输入用户名：</td>
                <td><input type="text" name="name" /></td>
        </tr>
        <tr>
                <td align="right">请输入密码：</td>
                <td><input type="password" name="pwd" /></td>
        </tr>
        <tr>
                <td colspan="2" align="center"><input type="submit" value="登  录" /></td>
        </tr>
       </table>
</form>
```

（3）新建 login.jsp 页，验证用户登录信息，关键代码如下：

```
<%
        String name = request.getParameter("name");
        String pwd = request.getParameter("pwd");
%>
<!-- 使用 useBean 动作标签导入 JavaBean 对象 -->
<jsp:useBean id="strBean" class="com.lh.bean.StringUtil"></jsp:useBean>
<!-- 对 StringUtil 类的属性赋值 -->
<jsp:setProperty property="name" name="strBean" value="<%=name %>"/>
<jsp:setProperty property="pwd" name="strBean" value="<%=pwd %>"/>
<table>
        <tr>
                <td align="right">提示信息：</td>
                <td>
                        <!-- 从 StringUtil 对象中获得属性值 -->
                        <jsp:getProperty property="cue" name="strBean"/>
```

```
            </td>
        </tr>
        <tr>
            <td align="center" colspan="2"><a href="index.jsp">[返回]</a></td>
        </tr>
</table>
```

秘笈心法

读者可以根据本实例的实现方法，实现网上办公自动化系统以及网上购物系统中的用户登录信息提示。

实例 149 输出分页导航的方法

光盘位置：光盘\MR\05\149

初级

实用指数：★★★★

实例说明

在实际开发过程中，很多的功能模块显示数据的部分都需要分页显示，而且分页部分的 HTML 代码都是相同的，为了提高开发效率以及便于维护，可以将这部分分页导航的代码封装在 JavaBean 中。本实例将介绍如何将分页导航的 HTML 代码封装到 JavaBean 中。

关键技术

本实例主要根据用户单击超链接时传递的当前页的参数来判断是否显示“上一页”或“下一页”的超链接。当单击“首页”时该参数值为 1，此时应该不存在“上一页”的超链接，只有“下一页”和“尾页”的超链接；当单击尾页时该参数值为总页数，应该没有“下一页”的超链接，只有“首页“和“上一页”的超链接；当单击“上一页”时，当前参数值应该减 1，单击“下一页”超链接时，当前值应该加 1。所以控制好当前页的参数值对实现分页导航是很关键的。

设计过程

新建名为 Page 的 JavaBean 类，该类用于封装分页导航的代码，关键代码如下：

```
public class Page {
private int pageSize = 10;                    //每页显示的记录数
private int currentPage = 1;                  //当前页
private int totalPage = 0;                    //总页数
private int totalRows = 0;                    //总记录数
private boolean hasBefore = false;            //是否有上一页
private boolean hasNext = false ;             //是否有下一页
private String linkHTML="";                   //用于保存分页导航的 HTML 代码
private String pageURL;                       //具体的链接地址
public int getTotalPage() {
        totalPage = ((totalRows + pageSize) - 1) / pageSize;    //根据数据总数和每页显示的记录数算出总页数
        return totalPage;
}
//单击的是首页
public void firstPage(){
        currentPage = 1;                      //当前页的值为 1
        this.setHasBefore(false);             //没有上一页
        this.refresh();                       //单击“首页”时应该设置是否有上一页和下一页
}
//单击的是上一页
public void beforePage(){
        currentPage -- ;                      //当前页的值减 1
        this.refresh();                       //单击“上一页”时应该设置是否有上一页和下一页

}
//单击的是下一页
public void nextPage(){
        if(currentPage < totalPage){
```

```
            currentPage ++ ;                    //当前页的值加 1
        }
        this.refresh();                         //单击“下一页”时应该设置是否有上一页和下一页
    }
    //单击的是尾页
    public void lastPage(){
        currentPage = totalPage ;               //当前页的值等于总页数
        this.setHasNext(false);                 //没有下一页
        this.refresh();                         //单击“尾页”时应该设置是否有上一页和下一页
    }
    //根据用户的操作，判断是否有上一页和下一页
    public void refresh(){
        if(totalPage<=1){                       //总页数小于等于 1 的情况，没有上一页和下一页
            this.setHasBefore(false);
            this.setHasNext(false);
        }
        else if(currentPage==1){                //当前页为首页，没有上一页，有下一页
            this.setHasBefore(false);
            this.setHasNext(true);
        }
        else if(currentPage==totalPage){        //当前页为尾页，没有下一页，有上一页
            this.setHasBefore(true);
            this.setHasNext(false);
        }else{                                  //除了以上的所有条件，有上一页和下一页
            this.setHasBefore(true);
            this.setHasNext(true);
        }
    }
    //获得分页导航代码的方法，主要根据是否有上一页和下一页来判断
    public String getLinkHTML() {
        linkHTML += "共"+this.totalRows+"条记录      ";
        if( this.hasBefore ){                   //如果有上一页，添加上一页的超链接代码
            linkHTML += "<a href='"+this.pageURL+"?currPage=1'>首页</a>";
            linkHTML += "    ";
            linkHTML += "<a href='"+this.pageURL+"?currPage="+this.currentPage+"&action=before'>上一页</a>";
            linkHTML += "    ";
        } else{                                 //如果没有上一页
            linkHTML += "首页        上一页    ";
        }
        if( this.hasNext ){                     //如果有下一页，添加下一页的超链接代码
            linkHTML += "<a href='"+this.pageURL+"?currPage="+this.currentPage+"&action=next'>下一页</a>";
            linkHTML += "    ";
            linkHTML += "<a href='"+this.pageURL+"?currPage="+this.totalPage+"'>尾页</a>";
            linkHTML += "    ";
        } else {                                //没有下一页
            linkHTML += "下一页        尾页    ";
        }
        linkHTML +="当前为"+this.currentPage+"/"+this.totalPage+"页";
        return linkHTML;
    }
}
```

秘笈心法

在本实例中实现将数据分页的代码封装到 JavaBean 中，避免了在 JSP 页中插入大量的 Java 脚本代码而导致 JSP 页中的代码混乱，而且有利于程序的维护管理。

实例 150　版权信息的生成方法

初级

光盘位置：光盘\MR\05\150

实用指数：★★★★

实例说明

在网站开发过程中，在每个页面的底部都需要添加网站的版权信息以便于网站的维护管理，可以将这些版

权信息封装到 JavaBean 中，从而达到可重复利用的效果，实例运行效果如图 5.25 所示。



图 5.25　生成版权信息

关键技术

本实例的实现很简单，首先在 JavaBean 中定义好版权信息的字符串，然后在 JSP 页中使用<jsp:userBean>动作导入 JavaBean 对象，最后通过<jsp:getProperty>动作获得 JavaBean 中定义好的版权信息的字符串属性值。

设计过程

（1）新建名为 Copyright 的 JavaBean 类，关键代码如下：

```
public class Copyright {
 private String copyrightStr="CopyRight 2006 www.mingrisoft.com " +"吉林省明日科技有限公司<br>" +
                "服务热线：0431-84978981 84978982  传真：0431-84972266";
 public String getCopyrightStr() {
         return copyrightStr;
 }
}
```

（2）新建 index.jsp 页，用于显示版权信息，关键代码如下：

```
<table>
    <tr bgcolor="skyblue">
                <td align="center">生成的版权信息：</td>
    </tr>
    <tr>
                <td>
                        <jsp:useBean id="copyright" class="com.lh.bean.Copyright"></jsp:useBean>
                        <jsp:getProperty property="copyrightStr" name="copyright"/>
                </td>
        </tr>
</table>
```

秘笈心法

在开发网站的过程中，经常会在多个页面的结尾重复地写入网站版权信息的 HTML 代码，在本实例中将版权信息的代码封装到 JavaBean 中，有利于代码的重用与维护，并提高开发效率。

5.5　窗口与对话框

在网站开发过程中，经常需要对网站的某个功能的操作给出提示信息的对话框或者弹出一个固定大小窗口，以便于用户正确地使用网站。

实例 151	弹出提示对话框并重定向网页 光盘位置：光盘\MR\05\151	初级 实用指数：★★★★

实例说明

弹出提示对话框是网站中经常用到的，为了便于维护，可以将弹出对话框的 JavaScript 代码封装到 JavaBean 中。在本实例中，当用户提交注册信息表单时，会弹出相应的提示对话框，运行效果如图 5.26 所示。

关键技术

本实例主要通过 JavaBean 将整个弹出对话框以及页面重定向的 JavaScript 代码封装在一个字符串中，然后

在页面中通过<jsp:useBean>动作导入 JavaBean，使用<jsp:setProperty>动作设置弹出消息以及重定向的页面链接，最后通过<jsp:getProperty>动作获得定义弹出对话框代码的属性值。



图 5.26　弹出提示对话框并重定向网页

设计过程

（1）新建名为 PopDialog 的 JavaBean 类，该类用于封装弹出对话框的信息并重定向页面，关键代码如下：

```
public class PopDialog {
private String dialogStr;                              //弹出对话框的代码
private String message;                                //弹出的提示消息
private String url;                                    //重定向的链接地址
public String getDialogStr() {
        dialogStr = "<script language ='javascript'>\r\n\t";
        dialogStr +="alert('"+message+"');\r\n\t";     //弹出对话框
        dialogStr +="window.location.href = '"+url+"';\r\n";   //页面重定向
        dialogStr += "</script>";
        return dialogStr;
}
……      //此处省略了其他属性的 get 和 set 方法
}
```

（2）在弹出对话框的页面引用以上定义的 JavaBean，关键代码如下：

```
<jsp:useBean id="popdialog" class="com.lh.bean.PopDialog"></jsp:useBean>
<jsp:setProperty property="message" name="popdialog"   value="注册成功！"/>
<jsp:setProperty property="url"   name="popdialog" value="index.jsp"/>
<jsp:getProperty property="dialogStr" name="popdialog"/>
```

秘笈心法

由于在 JSP 中并没有提供弹出提示对话框的函数，但在程序开发时经常需要弹出提示对话框，并在确定后重定向页面，因此实现本实例的功能是有必要的。

实例 152	打开指定大小的新窗口 光盘位置：光盘\MR\05\152	初级 实用指数：★★★★

实例说明

在实际开发过程中经常会用到弹出的窗口，为了便于维护管理以及提高代码的重用性，可以将弹出窗口的 JavaScript 代码封装在 JavaBean 中。本实例将讲解如何将弹出窗口的 JavaScript 代码封装到 JavaBean 中，其运行效果如图 5.27 所示。



图 5.27　弹出指定大小的窗口

关键技术

使用 JavaScript 打开一个弹出窗口，可以使用 window 对象的 open()方法或 showModalDialog()方法完成。showModalDialog()方法的示例代码如下：

```
var url="window1.jsp";
window.showModalDialog(url,'window','status=no;dialogWidth=560px;dialogHeight=190px;center=yes;help=no;location=no;');
```

open()方法的示例代码如下：

```
var url="window1.jsp";
window.open(url,'window','status=no,width=500px,height=100px,center=yes,help=no,location=no,menubar=no,toolbar=no');
```

设计过程

（1）新建名为 ShowWindow 的 JavaBean 类，用于封装 JavaScript 代码的打开指定大小的窗口的方法，关键代码如下：

```
public class ShowWindow {
private String url;                              //打开窗口的链接地址
private String openWindowStr="";                 //用于保存打开窗口的 JavaScript 代码
private int width;                               //打开窗口的宽度
private int height;                              //打开窗口的高度
private String functionName;                     //打开窗口的 JavaScript 函数名
public String getOpenWindowStr() {
        StringBuffer sb = new StringBuffer(openWindowStr);
        sb.append("<script language='javascript'>");
        sb.append("\r\n\t");                       //添加换行缩进
        sb.append("function "+this.functionName+"(){");    //添加函数名
        sb.append("\r\n\t\t");
        //打开一个窗口时，返回一个 window 类型的对象 returnObj，可以根据此对象来调整窗口的位置
        sb.append("var returnObj = window.open('"+this.url+"','window','width="+this.width+"px,height="+this.height+"px');");
        sb.append("\r\n\t\t");
        //screen 对象表示屏幕，此处设置相对于屏幕的 x 坐标
        sb.append("var x=(screen.width-"+width+")/2;");     //此处设置相对于屏幕的 x 坐标
        sb.append("\r\n\t\t");
        sb.append("var y=(screen.height-"+height+")/2;");   //此处设置相对于屏幕的 y 坐标
        sb.append("\r\n\t\t");
        sb.append("returnObj.moveTo(x,y);");             //调用 moveTo()方法改变窗口位置
        sb.append("\r\n\t}");
        sb.append("\r\n");
        sb.append("</script>");
        return sb.toString();
}
……          //此处省略了其他属性的 get 和 set 方法
}
```

（2）新建 index.jsp 页，在该页中使用<jsp:useBean>导入打开窗口的 JavaBean 对象，然后使用<jsp:setProperty>动作为该 JavaBean 对象中的属性赋值，最后通过<jsp:getProperty>动作获得打开窗口的属性值，关键代码如下：

```
<!-- 导入打开窗口的 JavaBean 类 -->
<jsp:useBean id="myWindow" class="com.lh.bean.ShowWindow"></jsp:useBean>
<!-- 设置打开窗口的 JavaScript 函数名 -->
<jsp:setProperty property="functionName" name="myWindow" value="openWindow1"/>
<!-- 设置打开窗口的 url 地址 -->
<jsp:setProperty property="url" name="myWindow" value="window.jsp"/>
<!-- 设置打开窗口的宽度 -->
<jsp:setProperty property="width" name="myWindow" value="200"/>
<!-- 设置打开窗口的高度 -->
<jsp:setProperty property="height" name="myWindow" value="100"/>
<!-- 获得打开窗口的 JavaScript 函数字符串 -->
<jsp:getProperty property="openWindowStr"   name="myWindow" />
<form action="window.jsp" method="post">
        <input   type="button" value="打开窗口" onclick="openWindow1()"/>
</form>
```

秘笈心法

使用 showModalDialog()方法弹出的窗口是模态窗口，不能最小化，只有关闭当前弹出的模态窗口才能操作其他页面。而使用 open()方法弹出的更像是一个页面，此时对操作其他页面没有影响，但是在弹出该页面之前会首先弹出一个警告对话框，从而会给网页浏览者带来不必要的操作。所以读者可以根据实际需求来选择使用这两个方法。

5.6　对数据库操作的 JavaBean

在开发程序的过程中经常会对数据库进行操作，为了提高代码的重用性，可以将对数据库的操作封装到一个单独的类中。这样既提高了代码的重用性，又有利于程序的维护。

实例 153	连接数据库的方法 光盘位置：光盘\MR\05\153	初级 实用指数：★★★★

实例说明

本实例主要是获得一个数据库连接。运行本实例，如图 5.28 所示，单击“获得连接”按钮，若数据库连接成功，则提示“获取连接成功”，否则提示“获取连接失败”。

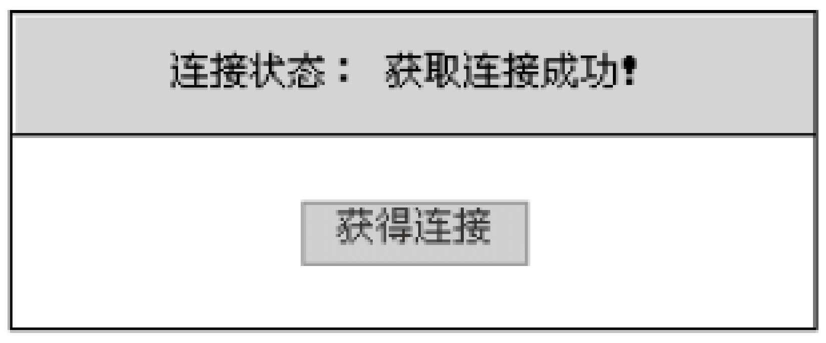


图 5.28　获得数据库的连接

关键技术

若要获得一个数据库连接，首先要加载数据库的驱动程序，然后再使用 DriverManager.getConnection()方法来获得数据库连接，具体语法格式如下：

```
DriverManager.getConnection(url,"sa","")
```

功能：获取数据库连接实例对象。

参数说明

url：是数据库的访问路径信息。例如，连接 SQL Server 2005 的 URL 格式如下：

```
jdbc:sqlserver://localhost:1433;DatabaseName=db_javabean
```

在使用上述方法获得数据库连接之前要设置驱动数据库的类的路径，具体语法格式如下：

```
Class.forName(classname)
```

功能：装载数据库的驱动类。

参数说明

classname：数据库驱动类的路径，如 com.microsoft.sqlserver.jdbc.SQLServerDriver。

注意：在 DriverManager.getConnection()方法中有 3 个参数：url 为数据库的 URL 地址，"sa"为登录数据库服务器的用户名；若登录服务器的密码为空，用""表示。

设计过程

（1）为实现本实例，首先需要创建一个 DB.java 文件来处理数据库的连接，关键代码如下：

```
import java.sql.*;
public class DB {
private Connection con;
private Statement stm;                                    //用于执行 SQL 语句的 Statement 类的实例对象
private ResultSet rs;                                     //结果集对象
private String classname="com.microsoft.sqlserver.jdbc.SQLServerDriver";
private String url ="jdbc:sqlserver://localhost:1433;DatabaseName=db_database05";
public DB(){
    try{
        Class.forName(classname);                         //加载驱动程序
      }catch(ClassNotFoundException e){e.printStackTrace();}
}
public Connection getCon(){
    try{
        con=DriverManager.getConnection(url,"sa","");     //获得连接
      }catch(Exception e){
            e.printStackTrace(System.err);
```

```
            con=null;}
        return con;
    }
    public void closed(){
        try{
            if(con!=null)con.close();                    //关闭连接
        }catch(Exception e){e.printStackTrace();}
    }
```

（2）创建连接数据库的首页面 index.jsp，关键代码如下：

```
<form action="dogetcon.jsp">
<table>
 …
 <tr>
    <td align="center" height="60" valign="middle">
        <input type="submit" value="获得连接">
    </td>
 </tr>
</table>
</form>
```

（3）新建一个 dogetcon.jsp 页面，该页面调用 DB 类来获得一个数据库连接。dogetcon.jsp 页面的关键代码如下：

```
<%@ page contentType="text/html;charset=GBK"%>
<%@ page import="java.sql.*"%>
<jsp:useBean id="db" class="com.jb.db.DB" scope="page"/>
<%
    …
Connection con=db.getCon();                    //调用 DB 类中的 getCon()方法来获得一个连接
    …
%>
```

秘笈心法

对数据库的连接配置的连接 URL、数据库驱动类、数据库名和数据库密码也可以写在*.properties 属性文件中，然后应用 java.util.Properties 类来读取这些属性值，再进行创建数据库的连接。当需要更换数据库时，只修改属性文件的配置即可，而不必每次都修改数据库连接类的代码，这样更利于系统的维护管理。

实例 154 数据库查询的方法

光盘位置：光盘\MR\05\154

初级

实用指数：★★★★

实例说明

在开发程序时，经常需要将数据库中的信息显示在网页中以供用户浏览和选择，如图 5.29 所示。那么如何将数据库中的信息显示在页面上呢？本实例将解决这个问题。

用户名	性别	年龄	职务
yxq	男	20	软件设计
admin	女	21	会计
莫明	男	22	经理
许久	男	23	员工

查询全部数据

图 5.29　数据的查询

关键技术

通过调用 Statement 对象的 executeQuery()方法来查询数据表并得到一个查询结果集，语法结构如下：

```
ResultSet rs=executeQuery(String sql)
```

功能：执行查询语句并返回查询结果。

参数说明

❶ sql：此参数是要执行的 SQL 语句。例如，本实例中查询所有信息的 SQL 语句如下：

```
select *|字段名表 from 数据表名
```

❷ rs：此参数是 ResultSet 类的实例对象，它用于存储从数据库中查询的结果集。

设计过程

（1）首先要获得一个数据库连接，然后才能对数据表进行查询操作。在实例 153 中的 DB 类文件基础上加入如下方法来实现数据的查询，关键代码如下：

```
public Statement getStm(){
    try{
        con=getCon();
        stm=con.createStatement();
        }catch(Exception e){e.printStackTrace(System.err);}
    return stm;
}
public Statement getStmed(){
    try{
        con=getCon();
        stm=con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                                ResultSet.CONCUR_READ_ONLY);
    }catch(Exception e){e.printStackTrace(System.err);}
    return stm;
}
public ResultSet getAllRs(){
    String sql="select * from tb_user";
    try{
        stm=getStmed();
        rs=stm.executeQuery(sql);
        }catch(SQLException e){e.printStackTrace();}
    return rs;
}
```

（2）建立一个数据查询的首页面 index.jsp，关键代码如下：

```
<form name="searchform" method="post" action="dosearchall.jsp">
       <table>
        <tr align="center" valign="middle" bgcolor="#CCCCCC" height="22">
          <td>用户名</td>
          <td>性别</td>
          <td>年龄</td>
          <td>职务</td>
        </tr>
<!--   start   -->
<%
         ResultSet rs=(ResultSet)session.getAttribute("resultset");
         …
         while(rs.next()){
%>
            <tr align="center" valign="middle" height="22">
              <td><%=rs.getString("user_name") %></td>
              <td><%=rs.getString("user_sex") %></td>
              <td><%=rs.getInt("user_age") %></td>
              <td><%=rs.getString("user_job") %></td>
            </tr>
       <%}%>
<!--   end   -->
            <tr>
               <td align="center" colspan="4">
                 <input type="submit" name="searchall" value="查询全部数据">
               </td>
            </tr>
     </table>
     </form>
```

（3）建立接收 Form 表单的页面 dosearchall.jsp，在该页面中进行数据查询操作，关键代码如下：

```
<%@ page contentType="text/html;charset=GBK"%>
<%@ page import="java.sql.*" %>
<jsp:useBean id="db" class="com.jb.db.DB" scope="page"/>
<%
  ResultSet rs=db.getAllRs();                    //获得查询的结果集
  session.setAttribute("resultset",rs);          //将得到的结果集存储在 session 会话中
  response.sendRedirect("index.jsp");            //返回到 index.jsp 页面显示数据信息
%>
```

秘笈心法

对于数据库查询方法的代码，可以将它写在一个类中，如果要查询数据库中一个表的多条数据，可以定义一个与表对应的 JavaBean 类，JavaBean 类的属性与表的字段相对应，然后在查询方法中将查询结果集的每一行数据封装在 JavaBean 中，最后将这些封装数据添加到 java.util.List 集合中。数据库的查询方法只返回一个封装数据的 List 集合即可，然后在页面中只需要调用这个方法，从而避免在 JSP 页面中编写大量对数据库操作的 Java 代码。

实例 155 带参数的数据查询

光盘位置：光盘\MR\05\155

初级

实用指数：★★★★

实例说明

在浏览网页时，经常需要得到符合某个条件的所有信息。本实例使用带参数的查询方法来实现查询符合某一条件的所有信息。运行程序，在图 5.30 所示的页面中选择“查询类型”为“性别”，查询关键字为“男”后，单击“查询”按钮即可查询所有性别为“男”的记录，如图 5.31 所示。



用户名	性别	年龄	职务	资金
莫明	男	22	经理	100.0
admin	女	21	会计	100.0
yxq	男	20	软件设计	100.0
swallow	男	26	老师	100.0
雨飞	女	24	歌手	100.0
许久	男	23	员工	100.0
荣少天	男	23	经理	100.0
虚心高洁	男	23	作家	100.0
mr	女	23	总经理	100.0
寂寞流星	男	22	程序员	100.0

图 5.30　带参数的数据查询



友情提示！

查询结果如下：

用户名	性别	年龄	职务	资金
莫明	男	22	经理	100.0
yxq	男	20	软件设计	100.0
swallow	男	26	老师	100.0
许久	男	23	员工	100.0
荣少天	男	23	经理	100.0
虚心高洁	男	23	作家	100.0
mr	男	23	总经理	100.0
寂寞流星	男	22	程序员	100.0

图 5.31　查询结果

关键技术

本实例主要应用了 SQL 的 SELECT 语句从数据库中查询符合条件的记录，该 SQL 语句的具体语法格式如下：

```
select *|字段名表 from 数据表名 where 字段名=值
```

功能：查询符合条件的数据记录，代码如下：

```
select * from tb_user where username='yxq'
```

设计过程

（1）在 DB 类中定义 getPartRs(String subsql,Strig subsqlvalue)方法来查询符合条件的记录，并返回 ResultSet 类的结果集。getPartRs()方法的完整代码如下：

```
  …//省略了建立数据库连接和获得 Statement 对象的代码
public ResultSet getPartRs(String subsql,String subsqlvalue){
    if(subsql==null)
        subsql="";
    if(subsqlvalue==null)
        subsqlvalue="";
    String sql="select * from tb_user where "+subsql+" ='"+subsqlvalue+"'";          //生成 SQL 语句
    try{
        stm=getStmed();                                          //获取 Statement 对象
        rs=stm.executeQuery(sql);                                //执行 SQL 语句，获取结果集
        }catch(SQLException e){e.printStackTrace();}
    return rs;                                                   //返回结果集对象
}
```

（2）创建带参数查询的首页面 index.jsp，在该页面中包含一个下拉列表框用来显示查询的条件，一个文本框供用户输入所要查询的值，单击“查询”按钮进行表单提交。此页面的关键代码如下：

```
<form name="searchform" method="post" action="dosearchpart.jsp">
     <table>
      <tr bgcolor="lightgrey" height="25">
        <td align="center">用户名</td>
        <td align="center">性别</td>
        <td align="center">年龄</td>
        <td align="center">职务</td>
        <td align="center">资金</td>
      </tr>
    <%
      ResultSet rsall=db.getAllRs();                                   //获得数据表中的所有记录
      while(rsall.next()){                                             //循环显示出所有记录
    %>
      <tr>
        <td align="center"><%=rsall.getString("user_name")%></td>
        <td align="center"><%=rsall.getString("user_sex")%></td>
        <td align="center"><%=rsall.getInt("user_age")%></td>
        <td align="center"><%=rsall.getString("user_job")%></td>
        <td align="center"><%=rsall.getFloat("user_money")%></td>
      </tr>
    <%
      }
    %>
      <tr bgcolor="lightgrey">
        <td align="center" colspan="5">
        查询类型：
        <select name="subsql">
            <option value="user_name">用户名</option>
            <option value="user_sex">性别</option>
            <option value="user_age">年龄</option>
            <option value="user_job">职务</option>
        </select>
        <input type="text" name="subsqlvalue" size="17">
        <input type="submit" name="searchpart" value="查询" onclick="return check()">
        </td>
      </tr>
     </table>
     </form>
```

（3）创建接收 Form 表单的 dosearchpart.jsp 页面，在该页面中调用 DB 类中的 getPartRs()方法进行查询，关键代码如下：

```
<%@ page import="java.sql.*" %>
<jsp:useBean id="db" class="com.jb.db.DB" scope="page"/>
<%
  ResultSet rs=null;
  boolean mark=true;
  String mess="";
  String subsql=request.getParameter("subsql");                      //获取用户选择的查询类型
  String subsqlvalue=request.getParameter("subsqlvalue");            //获取用户输入的查询条件
  ……//省略部分代码
%>
  <table>
   <tr>
    <td align="center" colspan="5"><%=mess%></td>
   </tr>
   <tr bgcolor="lightgrey" height="20">
    <td align="center">用户名</td>
    <td align="center">性别</td>
    <td align="center">年龄</td>
    <td align="center">职务</td>
    <td align="center">资金</td>
   </tr>
   <%
        subsqlvalue=new String(subsqlvalue.getBytes("ISO-8859-1"));
        rs=db.getPartRs(subsql,subsqlvalue);                         //调用 getPartRs()方法进行带参数的查询操作
        if(!rs.next()){
   %>
```

```
        <tr>
          <td align="center" colspan="5">没有记录显示！</td>
        </tr>
<%
        }
        else{
            rs.previous();
            while(rs.next()){
%>
        <tr>
          <td align="center"><%=rs.getString("user_name")%></td>
          <td align="center"><%=rs.getString("user_sex")%></td>
          <td align="center"><%=rs.getInt("user_age")%></td>
          <td align="center"><%=rs.getString("user_job")%></td>
          <td align="center"><%=rs.getFloat("user_money")%></td>
        </tr>
<%
            }//while
        }
%>
</table>
```

秘笈心法

如果知道结果集 ResultSet 中每个字段数据的索引，在应用 ResultSet 对象的 getXXX()方法获取数据时，也可以用索引指定，索引值是从 1 开始的。例如，获取结果集中第二个字段的值，可以用 getXXX(2)这种写法。

实例 156 向数据表中插入数据的方法

初级

光盘位置：光盘\MR\05\156

实用指数：★★★★

实例说明

大多数网站都有用户注册功能，当用户填写注册信息（如图 5.32 所示），单击“添加”按钮后，服务器将用户输入的信息添加到数据库中，如图 5.33 所示。



图 5.32　数据的增加

用户名	性别	年龄	职务	资金
莫明	男	22	经理	100.0
admin	女	21	会计	100.0
yxq	男	20	软件设计	100.0
swallow	男	26	老师	100.0
雨飞	女	24	歌手	100.0
许久	男	23	员工	100.0
荣少天	男	23	经理	100.0
虚心高洁	男	23	作家	100.0
mr	男	23	总经理	100.0
寂寞流星	男	22	程序员	100.0

图 5.33　增加数据后的数据表

关键技术

实现数据的增加使用了 insert into 语句。该语句用于向数据库中插入数据记录，其语法格式有两种，下面分别进行介绍。

语法 1：

```
insert into 数据表名(字段 1,字段 2,…字段 n) values(值 1,值 2,…值 n)
```

语法 2：

若该语句中的字段与数据表中的字段的数目和结构都相同，则可以写成下面的语句：

```
insert into 数据表名 values(值 1,值 2,…值 n)
```

设置了增加数据的 SQL 语句后，还要使用 Statement 对象的 executeUpdate()方法通过执行该 SQL 语句来实

现向数据表中增加记录。该方法的语法格式如下：

```
executeUpdate(String sql)
```

功能：执行 SQL 更新语句。

参数说明

sql：为向数据库中增加数据的 SQL 语句。

设计过程

（1）在 DB 类中定义 insert()方法用于执行向数据库中插入记录的 SQL 语句。在该方法中用到了 getStmed()方法，可参见光盘中的相关代码。insert()方法的完整代码如下：

```
  …//省略了建立数据库连接和获得 Statement 对象的代码
public int insert(String sql){
    int num=-1;
    if(sql==null)sql="";
    try{
        stm=getStmed();
        num=stm.executeUpdate(sql);
    }catch(Exception e){e.printStackTrace();num=-1;}
    return num;
}
```

（2）创建接收 Form 表单数据的页面 doinsert.jsp，在该页面中设置了 SQL 语句并进行增加操作，关键代码如下：

```
<jsp:useBean id="db" class="com.jb.db.DB"/>
<html>
<head>
 <title>增加数据</title>
  <link rel="stylesheet" type="text/css" href="css/style.css">
</head>
<%
  String    mess="";
  String    username=request.getParameter("username");
  …//省略了获取其他属性的代码
  float     money=0;
  boolean mark=true;
  if(username==null||username.equals("")){
    mark=false;
    mess="<li>请输入<b>用户名！</b></li>";
  }
  …//省略了判断其他属性的代码
  if(mark){
    try{
        age=Integer.parseInt(userage);
    }catch(Exception e){mark=false;mess+="<li>输入的<b>年龄</b>不是数字！</li>";}
    try{
        money=Float.parseFloat(usermoney);
    }catch(Exception e){mark=false;mess+="<li>输入的<b>资金</b>不是数字！</li>";}
  }
  if(mark){
    username=new String(username.getBytes("ISO-8859-1"),"gbk");
    usersex=new String(usersex.getBytes("ISO-8859-1"),"gbk");
    userjob=new String(userjob.getBytes("ISO-8859-1"),"gbk");
    String sql="insert into tb_user values('"+username+"','"+usersex+"',"+age+",'"+userjob+"',"+money+")";
    int i=db.insert(sql);
    db.closed();
    if(i>0)
       response.sendRedirect("show.jsp");
    else
       mess="插入失败！";
  }
%>
  <table>
   <tr bgcolor="lightgrey">
    <td>友情提示！</td>
   </tr>
```

```
    <tr>
      <td align="center"><%=mess%></td>
    </tr>
  </table>
```

秘笈心法

在获取表单请求数据时，经常发生的问题就是中文乱码，如果页面的编码格式不统一，就会发生这样的问题。因此，在获取表单请求数据之前，首先确定页面的编码格式是否一致，如表单页的编码格式为 UTF-8，那么在获取表单请求的处理页或 Servlet 中，也应该设置编码格式为 UTF-8。

实例 157 数据修改的方法

初级

光盘位置：光盘\MR\05\157

实用指数：★★★★

实例说明

任何网站都需要维护和更新数据库中的数据，其中使用最频繁的是对数据的修改操作，如论坛中需要提供用户昵称和头像的修改或更新回帖的数量等。本实例主要实现的是修改数据库中的某条记录。如图 5.34 所示，在“用户名”文本框中输入需要更新基本信息的用户名并单击“更新该用户”按钮，如果输入的用户名存在，则跳转到 showupdate.jsp 页面，如图 5.35 所示。进行更新操作，更新后的结果如图 5.36 所示，否则显示提示信息。



用户名	性别	年龄	职务	资金
admin	女	21	会计	100.0
mr	男	23	总经理	100.0
swallow	男	26	老师	100.0
yxq	男	22	员工	100.0
寂寞流星	男	22	程序员	100.0
莫明	男	22	经理	100.0
虚心高洁	男	23	作家	100.0
许久	男	23	员工	108.0
雨飞	女	24	歌手	92.0
张三*	女	24	服务生	500.0
用户名:	yxq			更新该用户

图 5.34　更新前的用户信息

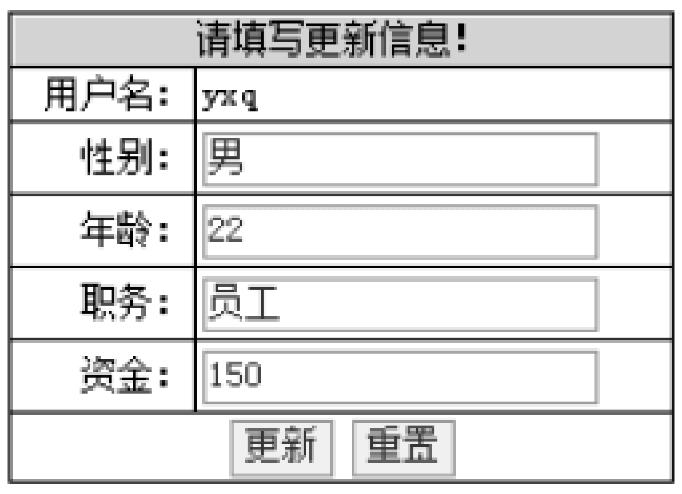


图 5.35　填写更新信息



用户名	性别	年龄	职务	资金
admin	女	21	会计	100.0
mr	男	23	总经理	100.0
swallow	男	26	老师	100.0
yxq	男	22	员工	150.0
寂寞流星	男	22	程序员	100.0
莫明	男	22	经理	100.0
虚心高洁	男	23	作家	100.0
许久	男	23	员工	108.0
雨飞	女	24	歌手	92.0
张三*	女	24	服务生	500.0
用户名:				更新该用户

图 5.36　更新后的用户信息

关键技术

本实例调用了 Statement 对象的 executeUpdate()方法执行 SQL 语句来实现数据的更新。实例中使用的 SQL 语句为更新（UPDATE）语句，语法格式如下：

```
UPDATE 数据表名 SET 字段名=值[,字段名 1=值 1,…,字段名 n=值 n]where 字段名=值
```

功能：更新数据表的数据。

设计过程

（1）在 DB 类中用定义 update()方法来执行更新数据表的 SQL 语句。在该方法中用到的 getStmed()方法的相关代码可参见光盘。update()方法的完整代码如下：

```
  …//省略了建立数据库连接和获得 Statement 对象的代码
public int update(String sql){
    int num=-1;
    if(sql==null)sql="";
      try{
          stm=getStmed();
          num=stm.executeUpdate(sql);
      }catch(Exception e){e.printStackTrace();num=-1;}
    return num;
}
```

（2）创建数据更新的首页面 index.jsp，该页面要求用户输入要修改的用户名，关键代码如下：

```
<form action="dosearch.jsp" name="searchuserform">
 <table>
  <tr bgcolor="lightgrey" height="25">
    <td align="center">用户名</td>
    <td align="center">性别</td>
    <td align="center">年龄</td>
    <td align="center">职务</td>
    <td align="center">资金</td>
  </tr>
  <%
    ResultSet rsall=db.getRs("select * from tb_user");                    //执行 SQL 语句，获取结果集
    while(rsall.next()){                                                  //循环结果集，显示数据表中的所有信息
  %>
  <tr>
    <td align="center"><%=rsall.getString("user_name")%></td>
    <td align="center"><%=rsall.getString("user_sex")%></td>
    <td align="center"><%=rsall.getInt("user_age")%></td>
    <td align="center"><%=rsall.getString("user_job")%></td>
    <td align="center"><%=rsall.getFloat("user_money")%></td>
  </tr>
 <%
    }
 %>
  <tr bgcolor="lightgrey">
    <td colspan="1" align="center">用户名：</td>
    <td colspan="3" align="cener">
       <input type="text" name="searchusername">
    </td>
    <td align="center" colspan="1">
       <input type="submit" name="searchsubmit" value="更新该用户"
                                  onclick="return searchcheck()">
    </td>
  </tr>
 </table>
 </form>
```

（3）创建处理 Form 表单的页面 dosearch.jsp。在该页面中查询输入的用户名是否存在，如果存在，跳转到 showupdate.jsp 页面；否则显示提示信息，关键代码如下：

```
<%@ page import="java.sql.*" %>
<jsp:useBean id="db" class="com.jb.db.DB"/>
<%
  String mess="";
  boolean mark=true;
  String username=request.getParameter("searchusername");               //获取用户输入的姓名
  if(username==null||username.equals("")){
    mark=false;
    mess="<li>请输入<b>用户名！</b></li>";
  }
  if(mark){
    username=new String(username.getBytes("ISO-8859-1"),"gbk");
    String sql="select * from tb_user where user_name='"+username+"'";   //按用户名查询表
    ResultSet rs=db.getRs(sql);                                          //获取查询结果集
    if(rs.next()){                                                       //遍历查询结果集
      session.setAttribute("searchusername",rs.getString("user_name"));  //将姓名保存在当前会话中
      db.closed();                                                       //关闭数据库连接
      response.sendRedirect("showupdate.jsp");                           //跳转到 showupdate.jsp 页面
    }
    else{
      db.closed();
      mess="您输入的用户名不存在！";
    }
  }
%>
  <table>
   <tr bgcolor="lightgrey">
    <td>友情提示！</td>
```

```
    </tr>
    <tr>
      <td align="center"><%=mess%></td>
    </tr>
  </table>
```

（4）创建供用户输入更新信息的页面 showupdate.jsp，关键代码如下：

```
<form action="doupdate.jsp" name="updateform">
 <table>
  <tr>
     <td colspan="2" align="center" bgcolor="lightgrey">
        请填写更新信息!
     </td>
  </tr>
  <tr>
    <td align="right">用户名：</td>
    <td><%=(String)session.getAttribute("searchusername")%></td>
  </tr>
  <tr>
    <td align="right">性别：</td>
    <td><input type="text" name="usersex" maxlength="1"></td>
  </tr>
  …//省略了其他用户信息的代码
  <tr>
    <td colspan="2" align="center">
        <input type="submit" name="submit" value="更新" onclick="return check()">
        <input type="reset" name="reset" value="重置">
    </td>
  </tr>
 </table>
</form>
```

（5）创建处理步骤（4）中 Form 表单的页面 doupdate.jsp。该页面用来接收表单数据，并进行更新操作。如果更新成功，则跳转到 index.jsp 页面；否则显示提示信息，关键代码如下：

```
<jsp:useBean id="db" class="com.jb.db.DB"/>
<%
  String    mess="";
  String    username=(String)session.getAttribute("searchusername");          //获取用户名
  String    usersex=request.getParameter("usersex");                          //性别
  String    userage=request.getParameter("userage");                          //年龄
  String    userjob=request.getParameter("userjob");                          //职位
  String    usermoney=request.getParameter("usermoney");                      //资金
  int       age=0;
  float     money=0;
  boolean mark=true;
  if(username==null||username.equals("")){
     mark=false;
     mess="<li>请输入<b>用户名！</b></li>";
  }
  …//省略了判断其他属性的代码
  if(mark){
     try{
          age=Integer.parseInt(userage);                                      //转换为整型
     }catch(Exception e){mark=false;mess+="<li>输入的<b>年龄</b>不是数字！</li>";}
     try{
          money=Float.parseFloat(usermoney);                                  //资金转换为浮点型
     }catch(Exception e){mark=false;mess+="<li>输入的<b>资金</b>不是数字！</li>";}
  }
  if(mark){
     usersex=new String(usersex.getBytes("ISO-8859-1"),"gbk");
     userjob=new String(userjob.getBytes("ISO-8859-1"),"gbk");
     String sql="update tb_user set user_sex='"+usersex+
                 "',user_age="+age+",user_job='"+userjob+
                 "', user_money="+money+" where user_name='"+username+"'";  //生成 SQL 语句
     int i=db.update(sql);                                                    //执行 SQL 语句
     db.closed();                                                             //关闭数据库连接
     if (i<=0)
          mess="更新失败！";
```

```
        else{
            session.invalidate();
            response.sendRedirect("index.jsp");
        }
    }
%>
    <table>
      <tr bgcolor="lightgrey">
        <td>友情提示！</td>
      </tr>
      <tr>
        <td align="center"><%=mess%></td>
      </tr>
    </table>
```

秘笈心法

在本实例实现的更新方法中，应用到的 Statement 接口可以改为 PreparedStatement 接口。应用 PreparedStatement 对 SQL 语句进行预编译，这样，在编写 SQL 语句时，就可以用“?”占位符来代替直接编写在 SQL 语句中的参数变量，然后再应用 PreparedStatement 的 setXXX()方法为参数进行赋值，这样可以避免 SQL 语句的注入式攻击。

实例 158	数据删除的方法 光盘位置：光盘\MR\05\158	初级 实用指数：★★★★

实例说明

本实例实现删除数据库中的某条记录的功能。运行本实例，在“用户名”文本框中输入 yxq，如图 5.37 所示，单击“删除该用户”按钮即可将该用户删除，删除后的结果如图 5.38 所示。



用户名	性别	年龄	职务	资金
莫明	男	22	经理	100.0
admin	女	21	会计	100.0
yxq	男	22	员工	150.0
swallow	男	26	老师	100.0
雨飞	女	24	歌手	100.0
许久	男	23	员工	100.0
荣少天	男	23	经理	100.0
虚心高洁	男	23	作家	100.0
mr	男	23	总经理	100.0
寂寞流星	男	22	程序员	100.0
用户名：	yxq		删除该用户	

图 5.37　删除前



用户名	性别	年龄	职务	资金
莫明	男	22	经理	100.0
admin	女	21	会计	100.0
swallow	男	26	老师	100.0
雨飞	女	24	歌手	100.0
许久	男	23	员工	100.0
荣少天	男	23	经理	100.0
虚心高洁	男	23	作家	100.0
mr	男	23	总经理	100.0
寂寞流星	男	22	程序员	100.0
用户名：			删除该用户	

图 5.38　删除后

关键技术

本实例主要调用 Statement 对象的 executeUpdate()方法执行 SQL 语句来实现数据的删除。实例中使用的 SQL 语句为 DELETE 语句，具体语法格式如下：

```
delete from  数据表名  where  字段名=值
```

功能：删除数据表中的某条记录。

设计过程

（1）在 DB 类中定义 delete()方法执行删除数据的 SQL 语句，完整代码如下：

```
…//省略了建立数据库连接的代码
public int delete(String sql){
    int num=0;
    if(sql==null)sql="";
```

```
        try{
            stm=getStmed();
            num=stm.executeUpdate(sql);
        }catch(Exception e){e.printStackTrace();num=-1;}
    return num;
}
```

（2）创建删除数据的首页面 index.jsp，关键代码如下：

```
<form action="dodelete.jsp" name="deleteform">
<table>
<tr bgcolor="lightgrey" height="25">
   <td align="center">用户名</td>
   <td align="center">性别</td>
   <td align="center">年龄</td>
   <td align="center">职务</td>
   <td align="center">资金</td>
</tr>
<%
   ResultSet rsall=db.getRs("select * from tb_user");                    //查询数据库中的记录
   while(rsall.next()){                                                  //显示数据表中的所有记录
%>
<tr>
   <td align="center"><%=rsall.getString("user_name")%></td>
   <td align="center"><%=rsall.getString("user_sex")%></td>
   <td align="center"><%=rsall.getInt("user_age")%></td>
   <td align="center"><%=rsall.getString("user_job")%></td>
   <td align="center"><%=rsall.getFloat("user_money")%></td>
</tr>
<%
   }
%>
<tr bgcolor="lightgrey">
   <td colspan="1" align="center">用户名：</td>
   <td colspan="3" align="cener">
       <input type="text" name="delusername">
   </td>
   <td align="center" colspan="1">
       <input type="submit" name="delsubmit" value="删除该用户" onclick="return check()">
   </td>
</tr>
</table>
</form>
```

（3）创建处理 Form 表单的页面 dodelete.jsp。该页面进行数据删除操作，如果删除成功，则跳转到 index.jsp 页面；否则显示提示信息，关键代码如下：

```
<%@ page import="java.sql.*" %>
<jsp:useBean id="db" class="com.jb.db.DB" scope="page"/>
<%
   boolean mark=true;
   String mess="";
   String username=request.getParameter("delusername");
   if(username==null||username.equals("")){
      mark=false;
      mess="<li>请输入<b>用户名！</b></li>";
   }
   if(mark){
      username=new String(username.getBytes("ISO-8859-1"),"gbk");
      String sql="select * from tb_user where user_name='"+username+"'";
      ResultSet rs=db.getRs(sql);
      if(!rs.next()){
          mess="输入的用户名不存在！";
      }
      else{
          sql="delete from tb_user where user_name='"+username+"'";          //生成 SQL 语句
          int num=db.delete(sql);                                            //调用执行 SQL 语句的方法
          db.closed();                                                       //关闭数据库连接
          if(num>0)
             response.sendRedirect("index.jsp");
```

```
            else
                mess="删除失败！";
        }
    }
%>
  <table>
    <tr>
      <td align="center"><%=mess%></td>
    </tr>
  </table>
```

秘笈心法

在实际应用中，一般都是根据 ID 来删除表数据的，因为只有 ID 是唯一的，其他字段值都有可能存在重复，这样可以避免误删除其他数据。本实例并没有根据表的 ID 值来删除数据，是由于此前设置了在添加用户信息时，用户名字段的值不能重复。读者可以对本实例稍作改动，应用表的 ID 来实现删除数据。

实例 159 数据分页的方法

光盘位置：光盘\MR\05\159

初级

实用指数：★★★★

实例说明

数据库分页是指通过查询语句从数据库中查询出某页所要显示的数据。本实例将从数据库中查询出每个页面需要显示的记录数。例如，某一数据表中有 10 条记录，若以每页 4 条记录来进行显示，在显示第二页信息时，只需查询从第 5 条开始到第 8 条的所有记录，实例运行结果如图 5.39 所示。



游戏名	人气	大小	上传者
《火拼泡泡龙》	极高	110MB	虚心高洁
《反恐精英》	高	200MB	潜入深蓝
《极品飞车》	高	202MB	忘忧草
《真三国无双》	高	89MB	柳柳
每页：4/10 条记录		当前页：2/3 页	
查询第一页			
第一页 上一页 下一页 最后一页			

图 5.39 数据库的分页显示

关键技术

由于本实例使用的是 SQL Server 数据库，而 SQL Server 对于查询分页的 SQL 语句主要应用的是 Top 关键字，具体代码如下：

```
select top m * from tb_game where id>(select MAX(id) from(select top (n-1)*m (id) from tb_game) as maxid)
```

功能：查询出只在当前页需要显示的所有记录。

参数说明

❶ n 为当前页码，m 为每页显示的记录数，id 是一个被设为自动编号的初始值为 1、增量为 1 的字段名。

❷ 子查询语句 1：“select top(n−1)*m (id) from tb_game”表示从 tb_game 表中查询出第 n 页前的所有记录。

❸ 子查询语句 2：“select MAX(id) from(select top (n−1)*m (id) from tb_game) as maxid”表示从子查询语句 1 中查询出字段 id 中的最大值。

设计过程

（1）在对数据库进行操作的 DB 类中定义如下属性和方法，关键代码如下：

```
private int num_per=4;                    //每页显示的记录数
private int num_rs=0;                     //总记录数
private int pages_all=0;                  //总页数
private int page_current=1;               //当前页数
  …//省略了建立数据库连接和获得 Statement 对象的代码
public void setPageInfo(int page){        //在该方法中计算出总记录数、总页数
                                          //并对通过参数传递的当前页数进行判断
    String sql="select * from tb_user";
    try{
        stm=getStmed();
        this.rs=stm.executeQuery(sql);
        this.rs.last();                   //将记录指针移动到最后一条记录
        this.num_rs=this.rs.getRow();     //获取当前指针所指记录的行号，即总记录数
```

```
        //计算出总页数
        this.pages_all=(num_rs%num_per==0)?(num_rs/num_per):(num_rs/num_per)+1
        if(page<1)                            //由参数传递过来的当前页数值如果小于 1，则将当前页数设置为 1
            this.page_current=1;
        else if(page>pages_all)               //由参数传递过来的当前页数值如果大于总页数，则将当前页数设置为总页数
            this.page_current=this.pages_all;
        else
            this.page_current=page;
    }catch(SQLException e){e.printStackTrace();}
}
public ResultSet getPageRs(){                 //数据库分页显示的关键代码
    int idnum=(this.page_current-1)*this.num_per;//第 n 页之前的记录数
    String sql="";
    if(this.page_current==1)                  //判断是否为第一页
        sql="select top "+this.num_per+" * from tb_game";
    else
        sql="select top "+this.num_per+" * from tb_game where id>(select MAX(id) from(select top "+idnum+" (id) from tb_game) as maxid)";
    try{
        rs=stm.executeQuery(sql);
    }catch(Exception e){e.printStackTrace();}
    return rs;
}
public int getNumper(){                       //返回每页显示记录数的方法
    return this.num_per;
}
public int getNumrs(){                        //返回总记录数的方法
    return this.num_rs;
}
public int getPagesall(){                     //返回总页数的方法
    return this.pages_all;
}
public int getPagecurrent(){                  //返回当前页数的方法
    return this.page_current;
}
```

（2）创建分页显示的首页面 index.jsp，关键代码如下：

```
<%@ page import="java.sql.*,com.jb.db.DB" %>
<% DB db=new DB(); %>
    <form name="searchform" method="post" action="dopagedb.jsp">
    <table>
     <tr align="center" valign="middle" bgcolor="#CCCCCC" height="22">
        <td>游戏名</td>
        <td>人气</td>
        <td>大小</td>
        <td>上传者</td>
     </tr>
     <%
        ResultSet rs=(ResultSet)session.getAttribute("pageresultset");
        if(rs==null){
     %>
     <tr align="center" valign="middle"><td colspan="4">没有记录显示！</td>
     </tr>
     <%
        }
        else{
            db=(DB)session.getAttribute("db");
            while(rs.next()){
     %>
     <tr align="center" valign="middle" height="22">
        <td><%=rs.getString("game_name") %></td>
        <td><%=rs.getString("game_hot") %></td>
        <td><%=rs.getString("game_size") %></td>
        <td><%=rs.getString("game_uper") %></td>
     </tr>
     <%
        }
     %>
     <tr bgcolor="lightgrey">
        <td colspan="4" align="center">
```

```
                每页：<%=db.getNumper()%>/<%=db.getNumrs()%> 条记录
                  
                当前页：<%=db.getPagecurrent()%>/<%=db.getPagesall()%> 页
            </td>
        </tr>
        <%
            }
        %>
        <tr>
            <td align="center" colspan="4">
        <input type="submit" name="searchall" value="查询第一页">
            </td>
        </tr>
        //以下为显示分页导航信息的代码
    <%if(db.getPagesall()>1){%>
            <tr bgcolor="lightgrey">
            <td align="center" colspan="4">
      <%if(db.getPagecurrent()>1){%>
        <a href="dopagedb.jsp?currentpage=1">第一页</a>
        <a href="dopagedb.jsp?currentpage=<%=db.getPagecurrent()-1%>">上一页</a>
      <%}%>
      <%if(db.getPagecurrent()<db.getPagesall()){%>
        <a href="dopagedb.jsp?currentpage=<%=db.getPagecurrent()+1%>">下一页</a>
        <a href="dopagedb.jsp?currentpage=<%=db.getPagesall()%>">最后一页</a>
      <%}%>
            </td>
        </tr>
      <%}%>
        </table>
        </form>
```

（3）创建处理 Form 表单的页面 dopagedb.jsp，关键代码如下：

```
<%@ page import="java.sql.*" %>
<jsp:useBean id="db" class="com.jb.db.DB" scope="page"/>
<%
    String strpage=request.getParameter("currentpage");        //获取当前要显示的页码
    if(strpage==null||strpage.equals(""))
        strpage="1";                                            //默认为第一页
    int currentpage=1;
    try{
        currentpage=Integer.parseInt(strpage);
        }catch(Exception e){currentpage=1;}
    db.setPageInfo(currentpage);
    ResultSet rs=db.getPageRs();                                //获取当前页面要显示的内容
    session.setAttribute("db",db);                              //将 db 对象保存到当前会话中
    session.setAttribute("pageresultset",rs);                   //将查询结果集保存到当前会话对象中
    response.sendRedirect("index.jsp");                         //跳转到 index.jsp 页面
%>
```

秘笈心法

在本实例的 dopagedb.jsp 页面中，将 db 对象放入 session 会话中是为了在返回到 index.jsp 页面后能够使用同一个 db 对象，而不是重新实例化一个 DB 类对象。如果想在不同 JSP 页之间使用同一个 JavaBean 实例，读者还可以在使用<jsp:useBean/>标签时将 scope 属性设置为 session 范围。

实例 160	对结果集进行分页的方法 光盘位置：光盘\MR\05\160	初级 实用指数：★★★★

实例说明

本实例讲解的是将用户查询到的所有的信息进行分页显示。例如，数据表中有 10 条记录，以每页 4 条记录进行显示，若要显示第二页的信息，首先应查询出所有的记录，然后在查询出的全部记录中查找要显示第二页

信息的开始位置。如图 5.40 所示，当用户单击“查询全部数据”按钮后，程序将查询出来的数据存储到 ResultSet 结果集中，然后单击“下一页”超链接，从该结果集中查找起始位置并显示出相应的信息。



图 5.40　数据库分页显示

关键技术

本实例根据当前的页数来确定结果集中记录指针的位置。该位置可以通过下面的算法得到：（当前页数-1）*每页显示的记录数+1。得到该位置后就可以使用 ResultSet 对象的 absolute(int pos)方法来移动记录指针至指定位置。

设计过程

（1）DB 类的关键代码如下：

```
private int num_per=4;                                  //每页显示的记录数
private int num_rs=0;                                   //总记录数
private int pages_all=0;                                //总页数
private int page_current=1;                             //当前页数
…//省略了建立数据库连接的代码
public Statement getStmed(){
    try{
        con=getCon();
        stm=con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);//只读模式
    }catch(Exception e){e.printStackTrace(System.err);}
    return stm;                                         //返回 Statement 对象
}
public void setPageInfo(int page){
    String sql="select * from tb_user";                 //声明 SQL 语句
    try{
        stm=getStmed();                                 //获取 Statement 对象
        this.rs=stm.executeQuery(sql);                  //执行 SQL 语句
        this.rs.last();                                 //将记录指针移动到最后一条记录
        this.num_rs=this.rs.getRow();                   //获取当前指针所指记录的行号，即总记录数
        //计算出总页数
        this.pages_all=(num_rs%num_per==0)?(num_rs/num_per):(num_rs/num_per)+1;
        if(page<1)                                      //由参数传递过来的当前页数值如果小于 1，则将当前页数设置为 1
            this.page_current=1;
        else if(page>pages_all)                         //由参数传递过来的当前页数值如果大于总页数，则将当前页数设置为总页数
            this.page_current=this.pages_all;
        else
            this.page_current=page;
    }catch(SQLException e){e.printStackTrace();}
}
public ResultSet getPageRs(){                           //对结果集进行分页显示的关键代码
   try{
       int pos=(this.page_current-1)*this.num_per+1;    //设置记录指针要移到的位置
       this.rs.absolute(pos);                           //将结果集记录指针移动到指定位置
   }catch(Exception e){e.printStackTrace();}
   return this.rs;
}
```

（2）创建分页显示的首页面 index.jsp，关键代码如下：

```
<form name="searchform" method="post" action="dopagers.jsp">
<table>
 <tr align="center" valign="middle" bgcolor="#CCCCCC" height="22">
    <td>用户名</td>
    <td>性别</td>
    <td>年龄</td>
    <td>职务</td>
 </tr>
 //如果 session 会话中不存在 ResultSet 对象
 <tr align="center" valign="middle">
    <td colspan="4">没有记录显示！</td>
 </tr>
```

```
//否则
<%
    db=(DB)session.getAttribute("db");
    int i=1;                                    //变量 i 用来控制显示的记录数
    rs.previous();                              //将指针移动到当前所指记录的前面，否则在下面调用 next()方法时将错过一条记录
    while(rs.next()&&i<=db.getNumper()){
%>
<tr align="center" valign="middle" height="22">
   <td><%=rs.getString("user_name") %></td>
   <td><%=rs.getString("user_sex") %></td>
   <td><%=rs.getInt("user_age") %></td>
   <td><%=rs.getString("user_job") %></td>
</tr>
<%
      i++;
    }
%>
…//此处省略了显示分页信息的代码
<tr>
   <td align="center" colspan="4">
      <input type="submit" name="searchall" value="查询全部数据">
   </td>
 </tr>
…//此处省略了显示分页导航信息的代码
</table>
</form>
```

（3）创建处理 Form 表单的页面 dopagers.jsp，关键代码如下：

```
<jsp:useBean id="db" class="com.jb.db.DB" scope="page"/>
<%
   String strpage=request.getParameter("currentpage");          //获取要显示的页码
   if(strpage==null||strpage.equals(""))strpage="1";
   int currentpage=1;
   try{
       currentpage=Integer.parseInt(strpage);
   }catch(Exception e){currentpage=1;}
   db.setPageInfo(currentpage);
   ResultSet rs=db.getPageRs();                                 //调用 getPageRs()方法来设置记录指针移至指定位置
   session.setAttribute("db",db);                               //将一个实例化的 DB 类对象放入 session 会话中
   session.setAttribute("pageresultset",rs);                    //将设置好的结果集放入 session 会话中
   response.sendRedirect("index.jsp");                          //返回到 index.jsp 页面
%>
```

秘笈心法

在本实例中，通过 Statement 对象的 executeQuery(String sql)方法得到的结果集必须是可滚动的。也就是说，在本实例中 Statement 对象是通过调用 DB 类中的自定义方法 getStmed()得到的，通过该方法得到的 Statement 对象才可以获得可滚动的结果集。

实例 161	关闭数据库的方法 光盘位置：光盘\MR\05\161	初级 实用指数：★★★★

实例说明

数据库连接是一个占用资源很大的操作，所以在对数据库的操作结束后，应及时关闭连接。本实例将介绍如何关闭数据库连接，实例运行效果如图 5.41 所示。

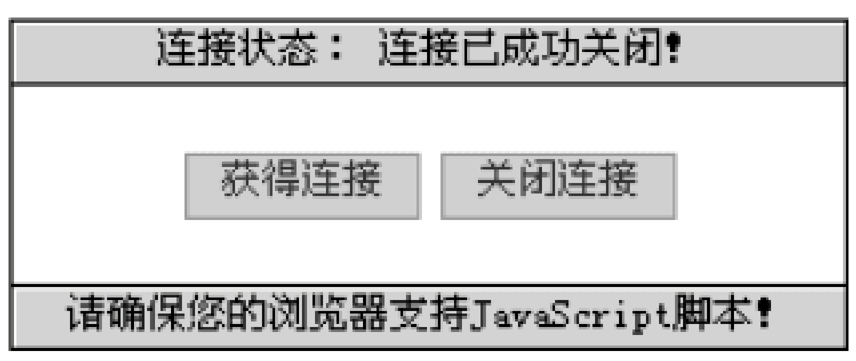


图 5.41　关闭数据库

关键技术

关闭数据库连接，主要调用的是 java.sql.Connection 接口的 close()方法。

关闭数据库连接的目的，就是为了释放数据库连接所占用的系统资源。

设计过程

（1）在对数据库进行操作的 DB 类中定义如下方法，关键代码如下：

```
…//省略了建立数据库连接和获得 Statement 对象的代码
  public void closed(){
      try{
          if(rs!=null)rs.close();            //关闭 rs 对象
      }catch(Exception e){e.printStackTrace();}
      try{
          if(stm!=null)stm.close();          //关闭 stm 对象
      }catch(Exception e){e.printStackTrace();}
      try{
          if(con!=null)con.close();          //关闭 con 对象
      } catch(Exception e){e.printStackTrace();}
  }
```

（2）创建关闭数据库的首页面 index.jsp，关键代码如下：

```
<script language="javascript">
  function change(){
     document.connectionform.action="doclose.jsp";
  }
</script>
  …
  <form action="dogetcon.jsp" name="connectionform">
  <table>
    <tr><td>此处为显示当前连接状态代码(省略)</td></tr>
    <tr>
      <td align="center" height="60" valign="middle">
        <input type="submit" name="create" value="获得连接">
        <input type="submit" name="close" value="关闭连接" onclick="change()">
      </td>
    </tr>
  </table>
  </form>
```

（3）创建一个用来关闭连接的页面 doclose.jsp，关键代码如下：

```
<jsp:useBean id="db" class="com.jb.db.DB" scope="page"/>
<%
    Connection con=(Connection)session.getAttribute("con");
    if(con!=null){
         db.closed();
         session.setAttribute("closestatus","连接已成功关闭！ ");
         session.removeAttribute("con");
    }
    else{
         session.setAttribute("closestatus","当前状态没有连接！ ");
    }
    session.setAttribute("formname","close");
    response.sendRedirect("index.jsp");
%>
```

秘笈心法

在程序开发中，当对数据库的操作结束后应及时调用该方法来关闭数据库连接，以释放所占用的系统资源。

实例 162	数据库事务处理的方法 光盘位置：光盘\MR\05\162	初级 实用指数：★★★★

实例说明

事务就是将多个操作封装到一个单元中，且该单元不可再分。该事务中的多个操作要么都被执行，要么都

不被执行。最常见的就是不同的账户间进行转账的例子，如图 5.42 所示，“雨飞”用户转账 50 元至“许久”用户，可通过两条更新语句来实现。若不使用事务处理，假设更新用户“雨飞”的操作成功，那么“雨飞”的账户上将减少 50 元，如果在更新“许久”用户的操作中出现错误，那么“许久”用户的账户上并没有加上从“雨飞” 账户上转过来的 50 元，这就失去了准确性。当出现上面的情况时，就可以用事务回滚的方法来撤销刚刚提交的事务，恢复到未操作之前的状态。

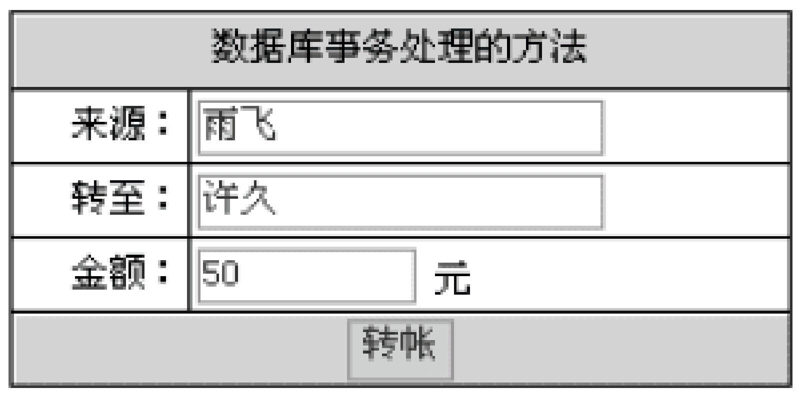


图 5.42　数据库事务处理

关键技术

当获得一个 Connection 对象后，调用它的 getAutoCommit()方法可得到一个 boolean 类型的值，该值默认为 true 表示自动提交事务。若要使用事务处理需调用 Connection 对象的 setAutoCommit(false)方法设置该值为 false，然后再调用 Connection 对象的 commit()方法来提交事务。若操作失败，就可调用 Connection 对象的 rollback()方法来回滚上次的操作。

设计过程

（1）在对数据库进行操作的 DB 类中定义 doback()方法，关键代码如下：

```
…//省略了建立数据库连接的代码
public String doback(String from,String to,float money){
   String mark="";
   …//此处为查询输入的用户名是否存在的代码
   fromsql="update tb_user set user_money=user_money-"+money+" where user_name='"+from+"'";
   tosql="update tb_user set user_money=user_money+"+money+" where user_name='"+to+"'";
   try{
        con=getCon();                          //获得一个连接
        con.setAutoCommit(false);              //设置为禁止自动执行
        stm=con.createStatement();
        stm.execute(fromsql);                  //执行 SQL 语句 1
        stm.execute(tosql);                    //执行 SQL 语句 2
        con.commit();                          //开始执行
        mark="success";                        //操作成功，返回 success
   }
   catch(Exception e){
        e.printStackTrace();
        mark="false";                          //操作失败，返回 false
        try{
             con.rollback();                   //回滚到上次操作之前
        }catch(Exception ee){ee.printStackTrace();}
   }
   return mark;
}
```

（2）创建数据库事务处理的首页面 index.jsp，关键代码如下：

```
<form action="dorollback.jsp" name="dorollbackform">
<table>
   <tr align="center" height="25" bgcolor="lightgrey">
      <td colspan="2" align="center">数据库事务处理的方法</td>
   </tr>
   <tr>
      <td align="right">来源：</td>
      <td><input type="text" name="from"></td>
   </tr>
   <tr>
      <td align="right">转至：</td>
      <td><input type="text" name="to"></td>
   </tr>
   <tr>
      <td align="right">金额：</td>
      <td><input type="text" name="money" size="10">
         元</td>
   </tr>
   <tr align="center" bgcolor="lightgrey">
```

```
        <td colspan="2">
            <input type="submit" name="Submit" value="转账" onclick="return check()">
        </td>
    </tr>
</table>
</form>
```

（3）创建接收 Form 表单的页面 dorollback.jsp。该页面调用 DB 类中的 doback()方法进行转账操作，关键代码如下：

```
<%@ page import="java.sql.*" %>
<jsp:useBean id="db" class="com.jb.db.DB"/>
<%
    boolean mark=true;
    String mess="";
    float money=0;
    String fromname=request.getParameter("from");
    String toname=request.getParameter("to");
    String strmoney=request.getParameter("money");
    if(fromname==null||fromname.equals("")){
        mark=false;
        mess+="<li>请输入<b>来源用户！ </b>";
    }
    …//省略了判断其他属性的代码
    if(mark){
        try{
            money=Float.parseFloat(strmoney);
        }catch(Exception e){mark=false;mess="输入的金额不是数字！ ";}
    }
    if(mark){
        fromname=new String(fromname.getBytes("ISO-8859-1"),"gbk");
        toname=new String(toname.getBytes("ISO-8859-1"),"gbk");
        String result=db.doback(fromname,toname,money);                    //调用事务处理
        if(result.equals("success"))
            mess="转账成功！ ";
        else if(result.equals("false"))
            mess="转账失败！ ";
        else
            mess=result;
    }
%>
```

秘笈心法

为了保证数据的统一完整性，应该在对数据库操作时及时应用事务来处理，保证在数据发生错误时取消操作，从而避免不必要的损失。

实例 163	调用数据库存储过程的方法 光盘位置：光盘\MR\05\163	初级 实用指数：★★★★

实例说明

存储过程是由存储在数据库管理系统中的一段 SQL 语句组成的一个过程。本实例通过向数据库中增加记录来讲解如何建立及使用存储过程，运行结果如图 5.43 所示。

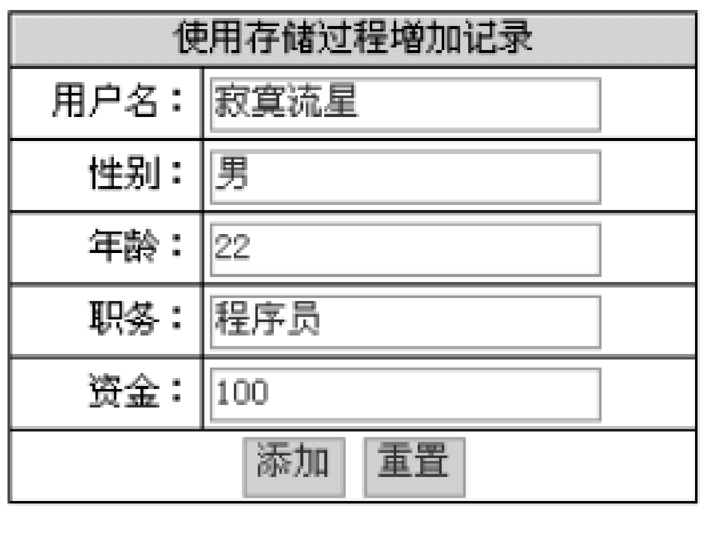


图 5.43　填写用户信息

关键技术

本实例主要通过 CallableStatement 对象来设置语句中的参数并执行调用。CallableStatement 对象是通过 Connection 对象的 prepareCall()方法得到的，语法格式如下：

```
prepareCall(String sql)
```

功能：获得 CallableStatement 对象。

参数说明

sql：表示调用存储过程的语句。

该方法将调用存储过程的对象 CallableStatement 与调用语句关联起来。其中调用语句的格式为{call 存储过程名(?,?,…,?)}，它是调用存储过程的通用格式。

设计过程

（1）首先在数据库服务器上新建一个增加记录的存储过程，关键代码如下：

```
CREATE PROCEDURE proc_add (
@username varchar(50),@usersex varchar(2),@userage int,@userjob varchar(50),@usermoney float)
AS
insert into tb_user values(@username,@usersex,@userage,@userjob,@usermoney)
GO
```

说明： 针对应用数据库的不同，其存储过程的语法结构可能存在差异，此存储过程的语法规则为 SQL Server 数据库中的写法。

（2）在对数据库进行操作的 DB 类中定义如下方法，关键代码如下：

```
…//省略了建立数据库连接和获得 Statement 对象的代码
public int useproc(String name,String sex,int age,String job,float money){
    int num=-1;
    String procsql="{call proc_add(?,?,?,?,?)}";          //调用存储过程
    try{
        con=getCon();
        CallableStatement stm=con.prepareCall(procsql);   //创建一个对象
        stm.setString(1,name);                           //设置调用语句中的参数
        stm.setString(2,sex);
        stm.setInt(3,age);
        stm.setString(4,job);
        stm.setFloat(5,money);
        num=stm.executeUpdate();                          //执行调用
     }
     catch(Exception e){e.printStackTrace();num=-1;}
     return num;
}
```

（3）调用存储过程向数据库中增加记录，其首页面设计与实例 162 的设计完全相同，读者可参看该实例的页面代码。

（4）创建接收首页面 index.jsp 中 Form 表单的页面 douseproc.jsp，关键代码如下：

```
<%@ page import="java.sql.*" %>
<jsp:useBean id="db" class="com.jb.db.DB"/>
<%
   String    mess="";
   String    username=request.getParameter("username");
   String    usersex=request.getParameter("usersex");
   String    userage=request.getParameter("userage");
   String    userjob=request.getParameter("userjob");
   String    usermoney=request.getParameter("usermoney");
   int       age=0;
   float     money=0;
   boolean mark=true;
   if(username==null||username.equals("")){
      mark=false;
      mess="<li>请输入<b>用户名！</b></li>";
   }
   …//省略了判断其他属性的代码
   if(mark){
     try{
        age=Integer.parseInt(userage);
     }catch(Exception e){mark=false;mess+="<li>输入的<b>年龄</b>不是数字！</li>";}
     try{
```

```
            money=Float.parseFloat(usermoney);
        }catch(Exception e){mark=false;mess+="<li>输入的<b>资金</b>不是数字！</li>";}
    }
    if(mark){
        username=new String(username.getBytes("ISO-8859-1"),"gbk");
        usersex=new String(usersex.getBytes("ISO-8859-1"),"gbk");
        userjob=new String(userjob.getBytes("ISO-8859-1"),"gbk");
        int i=db.useproc(username,usersex,age,userjob,money);
        db.closed();
        if(i<0){
            mess="操作失败！";
        }
        else
            mess="操作成功！";
    }
%>
```

秘笈心法

使用存储过程可以加快程序的执行速度，因为存储过程是预编译的，所以使用执行 SQL 语句的方法可节省 SQL 语句的编译时间；另外，还可以减少网络数据传输的信息量，因为存储过程是保留在数据库服务器中的。

第 6 章

Servlet 技术

- Servlet 基础
- Servlet 应用

6.1　Servlet 基础

Servlet 是 Java Web 应用中最核心的组件，也是 Web 服务器组件，它是一个中间控制层，负责处理客户端提交过来的请求数据以及调用业务逻辑对象，根据业务逻辑来控制页面转发。为了更好地理解 Servlet，本节将介绍一些 Servlet 基础方面的实例。

实例 164　动态生成 HTML 文档

光盘位置：光盘\MR\06\164

初级

实用指数：★★★★

实例说明

在实际开发过程中，有时需要直接在 Servlet 中将数据的处理结果反馈给客户端浏览器，这就需要通过 Servlet 来生成 HTML 内容。本实例将介绍如何在 Servlet 类中动态生成一个 HTML 页面，运行结果如图 6.1 所示。



图 6.1　使用 Servlet 动态生成的 HTML 页的内容

关键技术

本实例主要是通过在 Servlet 类中使用 HttpServletResponse 对象来动态生成 HTML 页。Servlet 通过该对象来生成响应结果，首先需要通过 HttpServletResponse 对象的 setContentType()方法来设置响应正文内容的 MIME 类型为 text/html，表示响应结果为 HTML 网页，然后再通过 getWriter()方法获得一个 PrintWriter 对象，该对象用于输出字符串形式的 HTML 正文数据。

设计过程

（1）新建名为 HTMLServlet 的 Servlet 类，该类继承自 HttpServlet 类，并在 doPost()方法中编写生成 HTML 文档的代码，关键代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        //设置响应的字符集格式为 UTF-8
        response.setCharacterEncoding("UTF-8");
        //设置响应正文的 MIME 类型
        response.setContentType("text/html");
        //返回一个 PrintWriter 对象，Servlet 使用它来输出字符串形式的正文数据
        PrintWriter out = response.getWriter();
        //以下为输出的 HTML 正文数据
        out.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\">");
        out.println("<HTML>");
        out.println("HEAD><TITLE>动态生成 HTML 文档</TITLE></HEAD>");
        out.println("<BODY>");
        out.println("<table border='0' align='center'>");
        out.println("<tr><td bgcolor='skyblue'colspan=2>动态生成 HTML 文档</td></tr>");
        out.println("</table>");
        out.println("</BODY>");
        out.println("</HTML>");
        out.flush();
        out.close();
}
```

（2）在 web.xml 中配置 HTMLServlet 类，此处修改配置的 URL 为“/html”，然后通过路径 http://localhost:8080/143/html 来访问该 Servlet 类，关键代码如下：

```
<servlet>
  <servlet-name>HTMLServlet</servlet-name>
  <servlet-class>com.lh.servlet.HTMLServlet</servlet-class>
</servlet>
<servlet-mapping>
```

```
    <servlet-name>HTMLServlet</servlet-name>
    <url-pattern>/html</url-pattern>
</servlet-mapping>
```

秘笈心法

在有些情况下，需要通过 Servlet 将数据处理的结果反馈给客户端浏览器，因此需要通过 Servlet 来生成 HTML 文档内容并响应给客户端，而且生成的 HTML 文档内容还可以包含 JavaScript 代码以及 CSS 样式等。

实例 165　在 Servlet 中实现页面转发

光盘位置：光盘\MR\06\165

初级

实用指数：★★★☆

实例说明

在实际的网站开发中，页面转发是最常见的。本实例将介绍如何在 Servlet 中控制页面的转发，在相应的文本框中输入用户名和密码，单击“登录”按钮，如果用户名和密码输入正确则页面转发到成功页，否则页面跳转到失败页，运行结果如图 6.2 所示。

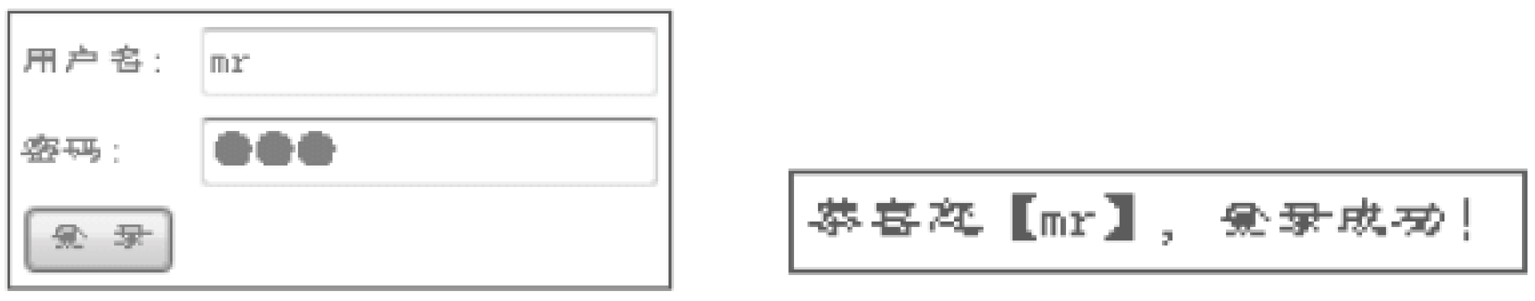


图 6.2　在 Servlet 中实现页面转发

关键技术

在 Servlet 中实现页面转发，使用的是 RequestDispatcher 对象的 forward()方法。可以在 Servlet 中通过 forward()方法将当前的请求转发到其他 Web 组件（如 Servlet、JSP、HTML）。

设计过程

（1）新建 index.jsp 页，该页主要包含一个输入用户名和密码的<form>表单，其中<form>中的 action 属性值为 Servlet 类在 web.xml 中配置的 URL，提交方式为 POST，关键代码如下：

```
<form action="forward" method="post">
    <table align="center">
        <tr>
            <td>用户名：</td>
            <td><input type="text" name="name" /></td>
        </tr>
        <tr>
            <td>密码：</td>
            <td><input type="password" name="pwd" /></td>
        </tr>
        <tr>
            <td colspan="2"><input type="submit" value="登 录" /></td>
        </tr>
    </table>
</form>
```

（2）新建名为 ForwardServlet 的 Servlet 类，在该类的 doPost()方法中获得表单请求信息，然后根据请求信息转发页面，关键代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    request.setCharacterEncoding("UTF-8");              //设置请求的字符编码格式
    String name = request.getParameter("name");         //获得请求表单中的用户名
    String pwd = request.getParameter("pwd");           //获得请求表单中的密码
    if((name!=null&&!name.equals(""))&&(pwd!=null&&!pwd.equals(""))){
        if(name.equals("mr")&&pwd.equals("123")){
```

```
                //使用 RequestDispatcher 对象将页面请求转发到 success.jsp 页
                request.getRequestDispatcher("success.jsp").forward(request, response);
            }else{
                request.getRequestDispatcher("error.jsp").forward(request, response);
            }
        }
}
```

（3）在 web.xml 中配置 ForwardServlet 类，主要修改用于配置 URL 的<url-pattern>标签值为“/forward”，关键代码如下：

```
<servlet>
  <servlet-name>ForwardServlet</servlet-name>
  <servlet-class>com.lh.servlet.ForwardServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>ForwardServlet</servlet-name>
  <url-pattern>/forward</url-pattern>
</servlet-mapping>
```

秘笈心法

如果实现的是单纯的页面跳转并不需要传递请求对象，可以使用 HttpServletResponse 对象的 sendRedirect()方法。如果想要在转发页面中获得当前的请求对象，那就需要通过 RequestDispatcher 对象的 forward()方法来实现。

实例 166　在 Servlet 中实现页面重定向

光盘位置：光盘\MR\06\166

初级　实用指数：★★★★

实例说明

本实例将介绍如何在 Servlet 中实现页面重定向。如图 6.3 所示，当用户输入的用户名或密码错误时，提交表单后在 Servlet 中会进行判断，然后将页面重定向到错误页。



图 6.3　在 Servlet 中实现页面重定向

关键技术

实现页面重定向主要应用的是 HttpServletResponse 对象的 sendRedirect()方法，该方法与页面转发的 forward()方法有本质的区别。使用 forward()方法时，会将当前正在处理的请求转发到其他 Web 组件（Servlet、JSP、HTML），如将请求转发到 index.jsp 页，在该页中可以通过 request 内置对象来获得此请求。而 sendRedirect()方法不会转发请求，只是简单的页面跳转。

设计过程

（1）新建名为 RedirectServlet 的 Servlet 类，该类继承自 HttpServlet 类，在该类的 doPost()方法中判断用户名和密码是否正确，如果不正确会使用 sendRedirect()方法将页面重定向到错误页，关键代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");          //设置请求数据的字符编码格式
        String name = request.getParameter("name");     //获得请求表单中的用户名
        String pwd = request.getParameter("pwd");       //获得请求表单中的密码
        if((name!=null&&!name.equals(""))&&(pwd!=null&&!pwd.equals(""))){
            if(name.equals("mr")&&pwd.equals("123")){
```

```
                //使用 RequestDispatcher 对象将页面请求转发到 success.jsp 页
                request.getRequestDispatcher("success.jsp").forward(request, response);
            }else{
                response.sendRedirect("error.jsp");//使用 sendRedirect()方法将页面重定向到 error.jsp
            }
        }
}
```

（2）在 Servlet 的配置文件 web.xml 中配置 RedirectServlet 类，关键代码如下：

```
<servlet>
        <servlet-name>RedirectServlet</servlet-name>
        <servlet-class>com.lh.servlet.RedirectServlet</servlet-class>
</servlet>
<servlet-mapping>
        <servlet-name>RedirectServlet</servlet-name>
        <url-pattern>/redirect</url-pattern>
</servlet-mapping>
```

秘笈心法

HttpServletResponse 对象的 sendRedirect()方法一律返回状态代码为 302 的响应结果，浏览器端接收到这种响应结果后，立即自动请求访问重定向的目标 Web 组件，客户端最后接收到的是目标 Web 组件的响应结果。

实例 167　在 Servlet 中处理表单提交的数据

光盘位置：光盘\MR\06\167

初级

实用指数：★★★★

实例说明

Java Web 的核心组件 Servlet 的主要功能是处理客户端的表单请求数据，在 Servlet 中首先对这些数据进行验证，可能会将其封装到 JavaBean，接下来调用数据库的业务逻辑方法将数据保存或者进行其他的操作，最后 Servlet 控制将响应结果返回给客户端。运行本实例，如图 6.4 所示，输入用户注册信息提交后，在 Servlet 中将获取这些表单请求信息，然后通过 forward()方法将这些请求信息转发，最后在转发页中显示出这些信息。



图 6.4　在 Servlet 中获取表单数据

关键技术

在 Servlet 中获取表单信息主要是应用 HttpServletRequest 对象的 getParameter()方法，该方法有一个 String 类型的参数，这个参数的名称与表单元素的 name 属性值对应，如果 getParameter()方法尝试读取一个 Null 值，执行时会抛出一个 java.lang.NULLPointerException 的异常，也就是说 getParameter()方法中的参数名必须要与表单元素中的 name 属性值一样，否则会发生异常。

例如，表单中有这样一个文本框，代码如下：

```
<input type="text" name="username" />
```

在 Servlet 中获取该表单文本框的值，代码如下：

```
String userName = request.getParameter("username");
```

在 Servlet 中会根据客户端表单<form>的 method 属性值来决定调用 doXXX()方法，如果 method 属性值为

POST，表单提交会调用 Servlet 中的 doPost()方法，如果 method 属性值为 GET，则调用 doGet()方法。

设计过程

（1）新建用户注册页 index.jsp，关键代码如下：

```
<form action="login" method="post">
        <table align="center">
                <tr>
                        <td>用户名：</td><td><input type="text" name="name" /></td>
                </tr>
                <tr>
                        <td>密码：</td><td><input type="password" name="pwd" /></td>
                </tr>
                <tr>
                        <td>性别：</td>
                        <td>
                                <input type="radio" name="sex" value="男" />男
                                <input type="radio" name="sex" value="女" />女
                        </td>
                </tr>
                <tr>
                        <td>年龄：</td><td><input type="text" name="age" /></td>
                </tr>
                <tr>
                        <td>Email：</td><td><input type="text" name="email" /></td>
                </tr>
                <tr>
                        <td colspan="2" align="center">
                                <input type="submit" value="注  册" /><input type="reset" value="重  置" />
                        </td>
                </tr>
        </table>
</form>
```

（2）新建名为 LoginServlet 的 Servlet 类，在 doPost()方法中获得表单请求信息，关键代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");          //设置请求的字符编码格式
        String name = request.getParameter("name");     //获得用户名
        String pwd = request.getParameter("pwd");       //获得密码
        String sex = request.getParameter("sex");       //获得性别
        String age = request.getParameter("age");       //获得年龄
        String email = request.getParameter("email");   //获得 Email
        request.getRequestDispatcher("logininfo.jsp").forward(request, response);
}
```

（3）在 web.xml 中配置 LoginServlet 类，关键代码如下：

```
<servlet>
      <servlet-name>LoginServlet</servlet-name>
      <servlet-class>com.lh.servlet.LoginServlet</servlet-class>
</servlet>
<servlet-mapping>
      <servlet-name>LoginServlet</servlet-name>
      <url-pattern>/login</url-pattern>
</servlet-mapping>
```

秘笈心法

在 Servlet 中获得客户端表单请求数据时，应该及时处理中文乱码问题，中文乱码问题一直是程序开发中不可避免的问题。有效的处理办法之一就是在 Servlet 中获取请求数据之前，应该通过 HttpServletRequest 对象的 setCharacterEncoding()方法统一请求信息的编码格式，当 JSP 页面的编码格式为 UTF-8 时，HttpServletRequest 对象的 setCharacterEncoding()方法的参数值也应该为 UTF-8。

实例 168

在 Servlet 中向客户端写 Cookie 信息

光盘位置：光盘\MR\06\168

初级

实用指数：★★★★

实例说明

本实例介绍的是如何在 Servlet 中向客户端写 Cookie 信息，运行本实例，如图 6.5 所示，输入用户名和密码并将其提交到 Servlet 中，在 Servlet 中将用户名添加到 Cookie 对象中，然后关闭浏览器，在重新访问用户登录页时，用户名的文本框中会显示上一次输入的用户名信息。



图 6.5　关闭浏览器再次访问的结果

关键技术

本实例的实现主要是应用 Servlet API 中提供的 Cookie 类。用户把表单信息提交给 Servlet 后，在 Servlet 中获取用户请求的信息并添加到 Cookie 对象中，再通过 HttpServletResponse 对象把 Cookie 信息返回给客户端，然后在 JSP 页面中通过 request 内置对象来获取客户端的 Cookie 信息。

在 JSP 中使用 request 对象获取的是一个 Cookie 对象的数组，需要在循环中遍历所有 Cookie 对象，并通过 Cookie 对象的 getName()方法查找所有 Cookie 对象的名称，然后根据找到的 Cookie 名称获得 Cookie 对象中的值。Cookie 类中包含的主要方法及说明如表 6.1 所示。

表 6.1　Cookie 类的主要方法及说明

方　法	说　明
getComment()/setComment(String purpose)	获取/设置 Cookie 的注释
getDomain()/setDomain(String pattern)	获取/设置 Cookie 适用的域。一般地，Cookie 只返回给与发送它的服务器名字完全相同的服务器
getMaxAge()/setMaxAge(int expiry)	获取/设置 Cookie 过期之前的时间，以秒为单位。如果不设置该值，则 Cookie 只在当前会话内有效，即在用户关闭浏览器之前有效
getName()/setName(String name)	获取/设置 Cookie 的名字
getValue()/setValue(String newValue)	获取/设置 Cookie 的值
getPath()/setPath(String uri)	获取/设置 Cookie 适用的路径。如果不指定路径，Cookie 将返回给当前页面所在目录及其子目录下的所有页面
getVersion()/setVersion(int v)	获取/设置 Cookie 所遵从的协议版本。默认版本为 0

设计过程

（1）新建用户登录表单页 index.jsp，关键代码如下：

```
<form action="cookieservlet" method="post">
    <table align="center">
            <tr><td>用户名：</td><td><input type="text" name="name" /></td></tr>
            <tr><td>密码：</td><td><input type="password" name="pwd" /></td></tr>
            <tr><td colspan="2"><input type="submit" value="登　录" /></td></tr>
        </table>
</form>
```

（2）新建名为 CookieServlet 的 Servlet 类，在该类的 doPost()方法中获取用户名信息，然后添加到 Cookie 对象中并保存到客户端，关键代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    request.setCharacterEncoding("UTF-8");
    String name = request.getParameter("name");                    //获得用户名
```

```
        name = java.net.URLEncoder.encode(name, "UTF-8");          //将用户名进行格式编码
        Cookie nameCookie = new Cookie("userName", name);          //创建一个 Cookie 对象，并将用户名保存到 Cookie 对象中
        nameCookie.setMaxAge(60);                                  //设置 Cookie 的过期之前的时间，单位为秒
        response.addCookie(nameCookie); //通过 response 的 addCookie()方法将此 Cookie 对象保存到客户端浏览器的 Cookie 中
        request.getRequestDispatcher("success.jsp").forward(request, response);
}
```

（3）在 index.jsp 页中读取所有客户端的 Cookie，通过循环 Cookie 数组找到保存用户名的 Cookie，关键代码如下：

```
<%
        String userName=null;                                      //用于保存从 Cookie 中读取出的用户名
        Cookie cookieArr[] = request.getCookies();                 //获取客户端的所有 Cookie
        if(cookieArr!=null&&cookieArr.length>0){
                for(Cookie c:cookieArr){
                        if(c.getName().equals("userName")){        //如果 Cookie 中有一个名为 userName 的 Cookie
                                userName = java.net.URLDecoder.decode(c.getValue(),"UTF-8");//将字符串解码，获得此 Cookie 的值
                        }
                }
        }
%>
```

（4）将获取到的用户名 Cookie 的值赋值给“用户名”文本框，关键代码如下：

```
<input type="text" name="name" value="<%if(userName!=null){out.print(userName);}%>"/>
```

（5）在 web.xml 文件中配置 CookieServlet 类，关键代码如下：

```
<servlet>
      <servlet-name>CookieServlet</servlet-name>
      <servlet-class>com.lh.servlet.CookieServlet</servlet-class>
</servlet>
<servlet-mapping>
      <servlet-name>CookieServlet</servlet-name>
      <url-pattern>/cookieservlet</url-pattern>
</servlet-mapping>
```

秘笈心法

在创建 Cookie 对象时，由于不可以直接将中文字符作为 Cookie 中的值，因此在将中文字符保存到 Cookie 对象之前，应该使用 java.net.URLEncoder 类的 encode()方法对中文字符进行编码。在获取该 Cookie 对象中的值时，需要使用 java.net.URLDecoder 类的 decode()方法对已经编码过的字符串进行解码，还原字符串的初始值。

实例 169	在 Servlet 中将 JavaBean 对象传递到 JSP 页 光盘位置：光盘\MR\06\169	初级 实用指数：★★★☆

实例说明

本实例将介绍如何将一个封装用户注册信息的 JavaBean 对象传递到 JSP 页中，然后在 JSP 页中读取该 JavaBean 对象中的数据，运行结果如图 6.6 所示。



图 6.6　在 JSP 中读取由 Servlet 传递过来的 JavaBean 对象中的数据

关键技术

实现本实例主要是在 Servlet 中使用 HttpServletRequest 对象的 getParameter()方法、setAttribute()方法以及

getAttribute()方法。setAttribute()方法的作用是可以在 HttpServletRequest 对象中保存一个属性，该方法的语法结构如下：

```
public void setAttribute(String name, Object object)
```

参数说明

❶ name：属性名。

❷ object：属性值。

设置完属性后，通过 getAttribute()方法来获取属性值，该方法的语法结构如下：

```
public Object getAttribute(String name)
```

根据参数 name 来查找请求范围内匹配的属性值。

设计过程

（1）新建用户注册页 index.jsp，关键代码如下：

```
<form action="passservlet" method="post">
	<table align="center">
		<tr>
			<td>用户名：</td>
			<td><input type="text" name="name" /></td>
		</tr>
		<tr>
			<td>密码：</td>
			<td><input type="password" name="pwd" /></td>
		</tr>
			… 此处省略了部分文本框的标签
		<tr>
			<td colspan="2" align="center">
				<input type="submit" value="注　册" />
				<input type="reset" value="重　置" />
			</td>
		</tr>
	</table>
</form>
```

（2）新建名为 UserInfo 的 JavaBean 类，该类用于封装用户的注册信息，关键代码如下：

```
public class UserInfo {
 private String userName;				//用户名
 private String userPwd;				//密码
 private String userSex;				//性别
 private int userAge;					//年龄
 private String email;					//电子邮件
 public UserInfo(){}		/			/默认的构造方法
…… //此处省略了属性的 getXXX()和 setXXX()方法
}
```

（3）新建名为 PassServlet 的 Servlet 类，在该类的 doPost()方法中将获取的用户注册信息封装到 UserInfo 中，然后将请求转发到 logininfo.jsp 页，关键代码如下：

```
 public void doPost(HttpServletRequest request, HttpServletResponse response)
			throws ServletException, IOException {
		request.setCharacterEncoding("UTF-8");		//设置请求的字符编码格式
		String name = request.getParameter("name");	//获取用户名
		String pwd = request.getParameter("pwd");	//获取密码
		String sex = request.getParameter("sex");	//获取性别
		String age = request.getParameter("age");	//获取年龄
		String email = request.getParameter("email");	//获取 Email
		UserInfo user = new UserInfo();			//创建封装用户信息的 JavaBean 对象
		//以下方法将获得的表单数据封装到 user 对象中
		user.setUserName(name);
		user.setUserPwd(pwd);
		user.setUserSex(sex);
		Integer userAge = new Integer(age);
```

```
        user.setUserAge(userAge);
        user.setEmail(email);
        request.setAttribute("User", user);                    //将 user 对象添加到 request 对象中
        request.getRequestDispatcher("logininfo.jsp").forward(request, response); //将请求转发到 logininfo.jsp 页面
}
```

（4）在 logininfo.jsp 页中，使用 request 内置对象的 getAttribute()方法获取封装用户注册信息的 JavaBean 对象，然后将该对象中封装的注册信息显示出来，关键代码如下：

```
<%
        UserInfo user = (UserInfo)request.getAttribute("User");
%>
<table align="center">
        <tr>
                <td>用户名：</td><td><%=user.getUserName()%></td>
        </tr>
        <tr>
                <td>密码：</td><td><%=user.getUserPwd()%></td>
        </tr>
        <tr>
                <td>性别：</td><td><%=user.getUserSex()%></td>
        </tr>
        <tr>
                <td>年龄：</td><td><%=user.getUserAge()%></td>
        </tr>
        <tr>
                <td>Email：</td><td><%=user.getEmail()%></td>
        </tr>
</table>
```

秘笈心法

本实例意在说明 MVC（Model-View-Controller）模式的三层架构设计理念，其中 JSP 相当于 View（视图）层、JavaBean 相当于 Model（模型）、Servlet 相当于 Controller（控制器）。MVC 架构有助于将应用程序分割成若干逻辑部件，使程序设计变得更加容易。基于 MVC 结构的开源框架包括 Struts1、WebWork、Struts2 等，而目前被广泛应用的框架是 Struts2，如果读者感兴趣，可以参考这些开源框架的相关资料或书籍。

实例 170　在 Servlet 中获取 Web 路径和文件真实路径

光盘位置：光盘\MR\06\170　　初级　　实用指数：★★★★☆

实例说明

在实际的开发中，经常需要获得 Web 路径以及某个文件的真实路径，然后根据这些路径来做相应的操作。本实例将介绍如何在 Servlet 中获取 Web 路径和真实路径。运行本实例，在浏览器的地址栏中输入“http://localhost:8080/149/getpath”，运行结果如图 6.7 所示。

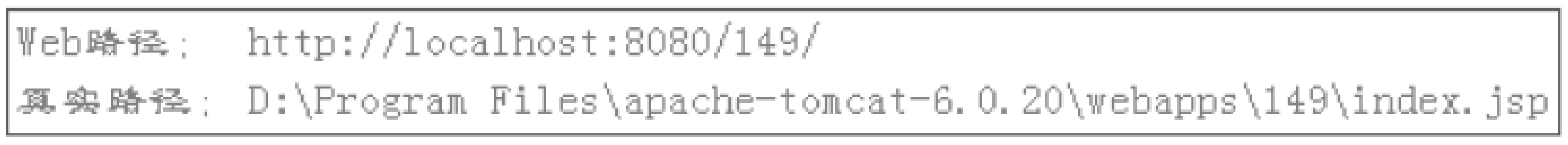


图 6.7　访问 Servlet 后输出的 Web 路径以及文件的真实路径

关键技术

在 Servlet 中，使用 HttpServletRequest 对象中的一系列方法可以获取相关路径的信息，然后可以根据这些信息组合成一个 Web 站点的虚拟路径。HttpServletRequest 接口中提供的用于获取路径有关的信息的方法如下。

- getScheme()：返回请求协议（http）。
- getServerName()：返回服务器的名称。如果访问本机，则返回的是localhost。
- getServerPort()：返回服务器的端口号。Tomcat服务器的默认端口为8080。

- getContextPath()：返回客户端所请求的Web应用的URL入口。例如，在浏览器中访问http://localhost:8080/helloapp/getpath，该方法返回的是“/helloapp”。
- getRequestURL()：返回请求Web服务器的完全路径。例如，在浏览器中访问http://localhost:8080/149/getpath，该方法就直接返回的是http://localhost:8080/149/getpath这个路径。

获取文件的真实路径，可以使用 ServletContext 接口的 getRealPath(String path)方法，该方法根据参数指定的虚拟路径，返回文件在系统中的真实路径，这个路径主要是指发布在 Tomcat 服务中的文件路径。例如，找出 index.jsp 文件的真实路径的写法为 getRealPath("index.jsp")，那么返回的真实路径为“D:\Program Files\apache-tomcat-7.0.41\webapps\149\index.jsp”。

设计过程

（1）新建名为 GetPathServlet 的 Servlet 类，在该类的 doPost()方法中获取 Web 服务器的路径以及文件的真实路径，关键代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        String scheme = request.getScheme();              //获取请求协议（http）
        String serverName = request.getServerName();  //获取服务器名称（localhost）
        int serverPort = request.getServerPort();         //获取服务器端口号
        String contextPath = request.getContextPath();  //返回 Web 应用的 URL 入口
        //以上这些属性组合成一个站点路径（http://localhost:8080/149/）
        String path = scheme+"://"+serverName+":"+serverPort+contextPath+"/";
        //getRealPath()方法返回一个给定虚拟路径的真实路径
        String realPath = this.getServletContext().getRealPath("index.jsp");
        //设置 HTTP 响应的正文的 MIME 类型以及字符编码格式
        request.setAttribute("path", path);               //将 Web 路径添加到 request 对象中
        request.setAttribute("realPath", realPath);       //将真实路径添加到 request 对象中
        request.getRequestDispatcher("path.jsp").forward(request, response);
}
```

（2）在 web.xml 中配置 GetPathServlet 类，具体代码请参考前面讲解的实例中给出的代码。

（3）新建 path.jsp 页，在该页中获得请求中保存的路径信息并显示，关键代码如下：

```
<table align="center">
        <tr>
                <td>Web 路径：</td><td><%=(String)request.getAttribute("path") %></td>
        </tr>
        <tr>
                <td>真实路径：</td><td><%=(String)request.getAttribute("realPath") %></td>
        </tr>
</table>
```

秘笈心法

在实际的开发过程中，经常需要根据获取的 Web 路径来实现文件的读写操作。例如，在实现将文件上传到服务器时，首先需要获取到 Web 服务器的工作目录，然后再根据该目录实现相应的操作。因此，读者应该掌握本实例的实现过程。

实例 171	在 Servlet 中访问 Web 应用的工作目录 光盘位置：光盘\MR\06\171	初级 实用指数：★★★☆

实例说明

本实例将介绍如何通过 Servlet 来访问 Web 应用的工作目录。运行本实例，在 Servlet 中读取 ServletContext 中的所有属性值，其中包含一个名为 javax.servlet.context.tempdir 的属性，该属性值代表当前 Web 应用的工作目录，运行结果如图 6.8 所示。

```
org.apache.catalina.WELCOME_FILES : [Ljava.lang.String;@dd20b6
javax.servlet.context.tempdir : D:\Program Files\apache-tomcat-6.0.20\work\Catalina\localhost\150
org.apache.catalina.jsp_classpath : /D:/Program%20Files/apache-tomcat-6.0.20/webapps/150/WEB-INF/classes/;/D:/Program%20Files/apache-tomcat-
6.0.20/webapps/150/WEB-INF/lib/jsf-api.jar;/D:/Program%20Files/apache-tomcat-6.0.20/webapps/150/WEB-INF/lib/jsf-impl.jar;/D:/Program%
20Files/apache-tomcat-6.0.20/webapps/150/WEB-INF/lib/jstl-1.2.jar;/D:/Program%20Files/apache-tomcat-6.0.20/lib/;/D:/Program%20Files/apache-tomcat-
6.0.20/lib/annotations-api.jar;/D:/Program%20Files/apache-tomcat-6.0.20/lib/catalina-ant.jar;/D:/Program%20Files/apache-tomcat-6.0.20/lib/catalina-
ha.jar;/D:/Program%20Files/apache-tomcat-6.0.20/lib/catalina-tribes.jar;/D:/Program%20Files/apache-tomcat-6.0.20/lib/catalina.jar;/D:/Program%
20Files/apache-tomcat-6.0.20/lib/el-api.jar;/D:/Program%20Files/apache-tomcat-6.0.20/lib/jasper-el.jar;/D:/Program%20Files/apache-tomcat-6.0.20/lib/jasper-
jdt.jar;/D:/Program%20Files/apache-tomcat-6.0.20/lib/jasper.jar;/D:/Program%20Files/apache-tomcat-6.0.20/lib/jsp-api.jar;/D:/Program%20Files/apache-
tomcat-6.0.20/lib/servlet-api.jar;/D:/Program%20Files/apache-tomcat-6.0.20/lib/tomcat-coyote.jar;/D:/Program%20Files/apache-tomcat-6.0.20/lib/tomcat-
dbcp.jar;/D:/Program%20Files/apache-tomcat-6.0.20/lib/tomcat-i18n-es.jar;/D:/Program%20Files/apache-tomcat-6.0.20/lib/tomcat-i18n-fr.jar;/D:/Program%
20Files/apache-tomcat-6.0.20/lib/tomcat-i18n-ja.jar;/D:/Program%20Files/apache-tomcat-6.0.20/bin/bootstrap.jar;/E:/Program%
20Files/MyEclipse/Common/binary/com.sun.java.jdk.win32.x86_1.6.0.013/lib/tools.jar;/E:/Program%
20Files/MyEclipse/Common/binary/com.sun.java.jdk.win32.x86_1.6.0.013/jre/lib/ext/dnsns.jar;/E:/Program%
20Files/MyEclipse/Common/binary/com.sun.java.jdk.win32.x86_1.6.0.013/jre/lib/ext/localedata.jar;/E:/Program%
20Files/MyEclipse/Common/binary/com.sun.java.jdk.win32.x86_1.6.0.013/jre/lib/ext/sunjce_provider.jar;/E:/Program%
20Files/MyEclipse/Common/binary/com.sun.java.jdk.win32.x86_1.6.0.013/jre/lib/ext/sunmscapi.jar;/E:/Program%
20Files/MyEclipse/Common/binary/com.sun.java.jdk.win32.x86_1.6.0.013/jre/lib/ext/sunpkcs11.jar
org.apache.catalina.resources : org.apache.naming.resources.ProxyDirContext@18eb00c
com.sun.faces.config.WebConfiguration : com.sun.faces.config.WebConfiguration@c7ecd5
org.apache.AnnotationProcessor : org.apache.catalina.util.DefaultAnnotationProcessor@18dabf1
```

图 6.8　打印 ServletContext 对象中所有的属性值

关键技术

在 Servlet 中，访问 Web 应用的工作目录主要是通过 ServletContext 对象的 getAttribute()方法，在 ServletContext 对象中包含一个名为 javax.servlet.context.tempdir 的属性，在 Servlet 初始化时会自动在 ServletContext 对象中设置该属性，通过这个属性返回的对象就是 Web 应用的工作目录。在 Servlet 中获得 Web 应用的工作目录的关键代码如下：

```
File workPath = (File)context.getAttribute("javax.servlet.context.tempdir");
```

设计过程

（1）新建名为 WebWorkPathServlet 的 Servlet 类，在该类的 doPost()方法中，获得所有的 ServletContext 中的属性并打印输出，然后获得 Web 应用的工作目录，并向该目录中写入一个文本文件，关键代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");     //响应正文的 MIME 类型以及字符编码格式
    PrintWriter out = response.getWriter();
    ServletContext context = this.getServletContext();
    Enumeration e = context.getAttributeNames();            //获得 ServletContext 对象中的所有属性名
    while(e.hasMoreElements()){                              //循环 Enumeration 对象，读出 ServletContext 对象中的所有属性值
      String attributeName = (String)e.nextElement();
      out.println("<br>"+attributeName+" : "+context.getAttribute(attributeName));
    }
    out.close();
    //获得 Web 应用的工作目录
    File workPath = (File)context.getAttribute("javax.servlet.context.tempdir");
    FileWriter writer =new FileWriter(workPath+"/test.txt");
    BufferedWriter bf = new BufferedWriter(writer);
    bf.write("获取 Web 应用的工作目录");                       //向文件中写入字符
    bf.newLine();                                            //换行
    bf.write("明日科技！ ");
    bf.flush();                                              //刷新缓冲区
    bf.close();                                              //关闭缓冲区输出流
}
```

（2）在 web.xml 中配置 WebWorkPathServlet 类，关键代码如下：

```
<servlet>
    <servlet-name>WebWorkPathServlet</servlet-name>
    <servlet-class>com.lh.servlet.WebWorkPathServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>WebWorkPathServlet</servlet-name>
    <url-pattern>/webworkpath</url-pattern>
</servlet-mapping>
```

秘笈心法

每个 Web 应用都有一个工作目录，Servlet 容器会把与这个 Web 应用相关的临时文件存放在这个目录下。

在 Servlet 中通过 ServletContext 对象来访问 Web 应用的工作目录。

6.2　Servlet 应用

本节将介绍一些在 Web 开发中常用的 Servlet 技术。例如，在 Servlet 中生成验证码、将表格数据导出到 Excel、避免客户端访问的并发问题以及访问数据库连接池等。

实例 172　记录用户访问次数

光盘位置：光盘\MR\06\172

初级　实用指数：★★★★☆

实例说明

在浏览网站时，有些网站会有计数器的功能，浏览者每访问一次网站，计数器就累加一次。运行本实例，当用户访问时，实现记录用户的访问次数，运行结果如图 6.9 所示。



图 6.9　记录用户的访问次数

关键技术

实现计数器主要是在 Servlet 中应用 ServletContext 接口，Servlet 容器在启动一个 Web 应用时，会为它创建一个 ServletContext 对象。当 Servlet 容器终止一个 Web 应用时，ServletContext 对象也会被销毁，所以该对象与 Web 应用程序有同样的生命周期。也就是说，整个 Web 应用的组件可以共享 ServletContext 对象中存放的共享数据。在 ServletContext 接口中存取共享数据的方法包括以下几种。

- setAttribute(String name,Object object)：在ServletContext对象中存放共享数据，参数name表示属性名，参数object表示属性值。
- removeAttribute(String name)：根据指定参数name属性名，删除ServletContext对象中的共享数据。
- getAttribute(String name)：根据指定的参数name属性，获取ServletContext对象中的共享数据。

设计过程

（1）新建名为 CounterServlet 的 Servlet 类，在该类的 doPost()方法中实现统计用户的访问次数，关键代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    ServletContext context = getServletContext();              //获得 ServletContext 对象
    Integer count = (Integer)context.getAttribute("counter");  //从 ServletContext 中获得计数器对象
    if(count==null){                                           //如果为空，则在 ServletContext 中设置一个计数器的属性
        count=1;
        context.setAttribute("counter", count);
    }else{                                                     //如果不为空，则设置该计数器的属性值加 1
        context.setAttribute("counter", count+1);
    }
    response.setContentType("text/html");                      //响应正文的 MIME 类型
    response.setCharacterEncoding("UTF-8");                    //响应的编码格式
    PrintWriter out = response.getWriter();
    out.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\">");
    out.println("<HTML>");
    out.println("  <HEAD><TITLE>统计网站访问次数</TITLE></HEAD>");
    out.println("  <BODY>");
    out.print("    <h2><font color='gray'> ");
    out.print("您是第  "+context.getAttribute("counter")+"  位访客！ ");
    out.println("</font></h2>");
    out.println("  </BODY>");
    out.println("</HTML>");
```

```
        out.flush();
        out.close();
}
```

（2）在 web.xml 中配置 CounterServlet 类，关键代码如下：

```
<servlet>
    <servlet-name>CounterServlet</servlet-name>
    <servlet-class>com.lh.servlet.CounterServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>CounterServlet</servlet-name>
    <url-pattern>/counter</url-pattern>
</servlet-mapping>
```

秘笈心法

在实际开发应用中，经常需要通过 ServletContext 对象来访问 Web 应用的各种资源。通过 ServletContext 对象可以实现以下操作：

（1）在 Web 应用范围内存取共享数据。

（2）访问当前 Web 应用的资源。

（3）访问 Servlet 容器中的其他 Web 应用。

（4）访问 Servlet 容器的相关信息。

（5）访问服务器端的文件系统资源。

（6）向 Servlet 的日志文件中输出日志。

实例 173 将数据导出到 Excel

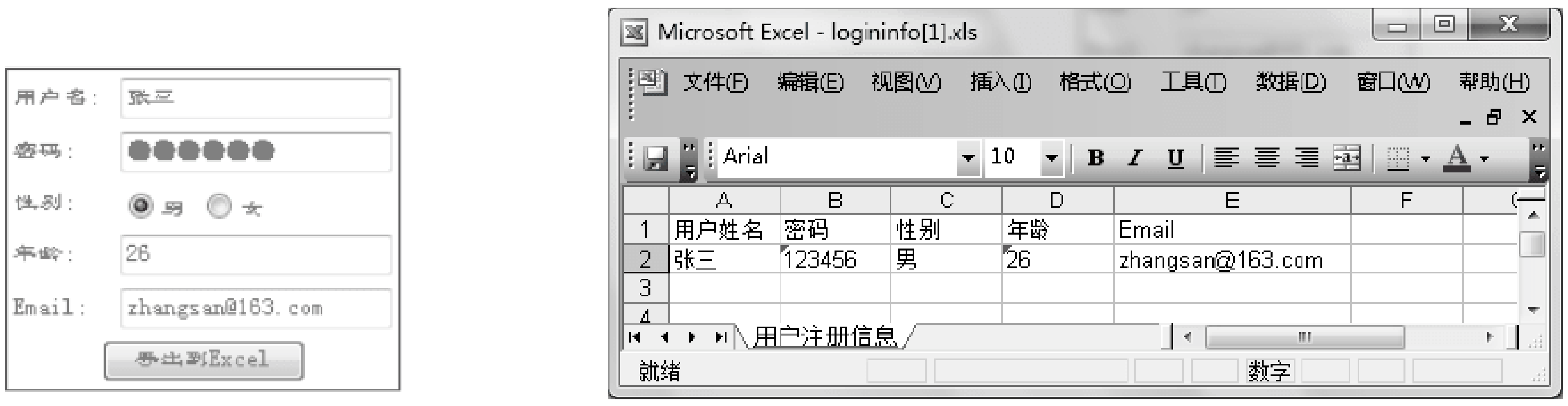
光盘位置：光盘\MR\06\173

初级

实用指数：★★★☆

实例说明

在实际的开发中，经常需要将一些数据导出到 Excel 或者 Word 来进行处理。本实例将介绍如何使用 POI 开源组件实现将数据导出到 Excel 文件中。运行本实例，输入用户注册信息后，单击“导出到 Excel”按钮后，会将用户的信息导出到 Excel 文件中，如图 6.10 所示。



（a）　　　　（b）

图 6.10　将用户注册的信息导出到 Excel 中

关键技术

实现本实例的关键使用的是开源的 POI 组件，该组件中包含实现对 Excel 文件的创建和写入操作的类。使用 POI 组件操作 Excel 文件的关键步骤如下：

（1）创建 Excel 的工作表。POI 组件的 HSSFWorkbook 类提供了创建工作表的方法，语法格式如下：

```
public HSSFSheet createSheet(String sheetname) //参数 sheetname 表示工作表的名称
```

（2）创建表格的行。在保存之前需要创建行对象，该对象由工作表对象创建，语法格式如下：

```
public HSSFRow createRow(int rownum) //参数 rownum 表示工作表中行对象的行号
```

（3）创建表格的单元格。Excel 表格中的数据由多个单元格组成，在 POI 组件中，这些单元格对象由 HSSF Row 类的 createRow()方法创建，语法格式如下：

```
public HSSFCell createCell(int columnIndex)//参数 columnIndex 表示单元格对象的列编号
```

（4）写入单元格内容。单元格对象定义了写入各种类型数据的方法，其中最常用的就是 String 类型的字符串数据。本实例使用最多的也是这个方法，语法格式如下：

```
public void setCellValue(String value)//参数 value 表示保存在 Excel 单元格中的数据
```

设计过程

（1）新建用户注册表单页 index.jsp，关键代码如下：

```
<form action="export" method="post">
	<table align="center">
		<tr>
			<td>用户名：</td><td><input type="text" name="name" /></td>
		</tr>
			…由于篇幅有限，此处省略了表单中的其他元素
		<tr>
			<td colspan="2" align="center">
				<input type="submit" value="导出到 Excel" />
			</td>
		</tr>
	</table>
</form>
```

（2）新建名为 ExportServlet 的 Servlet 类，在该类的 doPost()方法中获得用户注册信息，然后使用 POI 组件中的类将用户注册信息导出到 Excel 文件中，关键代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
		throws ServletException, IOException {
	request.setCharacterEncoding("UTF-8");
	response.setContentType("application/vnd.ms-excel");//响应正文的 MIME 类型，表示 Excel
	response.addHeader("Content-Disposition", "attachment;filename=logininfo.xls");
	String name = request.getParameter("name");
	String pwd =request.getParameter("pwd");
	String sex = request.getParameter("sex");
	String age = request.getParameter("age");
	String email = request.getParameter("email");
	ServletOutputStream out = response.getOutputStream();	//响应输出流对象
	HSSFWorkbook wb = new HSSFWorkbook();			//创建 Excel 表格
	HSSFSheet	sheet = wb.createSheet("用户注册信息");	//创建工作簿
	sheet.setColumnWidth(4, 5000);				//设置列宽
	HSSFRow titleRow = sheet.createRow(0);			//创建 Excel 中的标题行
	HSSFCell titleCell1 =titleRow .createCell(0);		//在行中创建第 1 个单元格
	titleCell1.setCellValue("用户姓名");			//设置第 1 个单元格的值
	HSSFCell titleCell2= titleRow.createCell(1);		//在行中创建第 2 个单元格
	titleCell2.setCellValue("密码");				//设置第 2 个单元格的值
	HSSFCell titleCell3 =titleRow .createCell(2);		//在行中创建第 3 个单元格
	titleCell3.setCellValue("性别");				//设置第 3 个单元格的值
	HSSFCell titleCell4= titleRow.createCell(3);		//在行中创建第 4 个单元格
	titleCell4.setCellValue("年龄");				//设置第 4 个单元格的值
	HSSFCell titleCell5= titleRow.createCell(4);		//在行中创建第 5 个单元格
	titleCell5.setCellValue("Email");			//设置第 5 个单元格的值
	HSSFRow valueRow = sheet.createRow(1);			//创建第 2 行
	HSSFCell nameCell = valueRow.createCell(0);		//在第 2 行中创建单元格
	nameCell.setCellValue(name);
	HSSFCell pwdCell = valueRow.createCell(1);
	pwdCell.setCellValue(pwd);
	HSSFCell sexCell = valueRow.createCell(2);
	sexCell.setCellValue(sex);
	HSSFCell ageCell = valueRow.createCell(3);
	ageCell.setCellValue(age);
	HSSFCell emailCell = valueRow.createCell(4);
	emailCell.setCellValue(email);
	HSSFCellStyle cellStyle = wb.createCellStyle();
	wb.write(out);						//将响应流输入到 Excel 表格中
```

```
        out.flush();
        out.close();
}
```

（3）在 web.xml 中配置 ExportServlet 类，具体代码请参考实例 172，只需要把实例 172 中的 CounterServlet 类换成本实例中的 ExportServlet 类即可。

秘笈心法

在 Servlet 中，通过 HttpServletResponse 对象的 setContextType()方法设置不同 MIME（Multipurpose Internet Mail Extension，多用途 Internet 邮件扩展）类型，Servlet 可以生成不同类型的响应文件。例如，Word 文档、Excel 文档、XML 文档、图像等。

实例 174 利用 Servlet 生成动态验证码

光盘位置：光盘\MR\06\174

高级

实用指数：★★★☆

实例说明

如今，绝大多数网站或者 Web 应用程序都实现了验证码的功能，加入验证码可以防止黑客利用恶意程序，在网站中进行频繁登录、注册、灌水等操作。运行本实例，在用户注册的表单中包含一个验证码图片和文本框，运行效果如图 6.11 所示。



图 6.11 利用 Servlet 生成的验证码

关键技术

在 Servlet 中，首先要设置响应正文的类型为 image/jpeg，表示响应的是一个图片，然后通过 java.awt 包中的操作图形图像的类来生成一个图像，主要用到以下几个类。

- java.awt.image.BufferedImage：创建该对象时，会在缓存中构造一个图像。
- java.awt.Graphics：该类的对象表示一个画笔，可以用它来画矩形、写字、添加颜色、画线等。
- java.util.Random：该类的对象用于生成随机数。本实例中使用它来生成图像上的随机坐标的值、随机颜色的值以及随机的验证码数字。

设计过程

（1）新建名为 ValidateCodeServlet 的 Servlet 类，在该类的 doPost()方法中实现生成验证码图片，关键代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
        //禁止页面缓存
        response.setHeader("Pragma", "No-cache");
        response.setHeader("Cache-Control", "No-cache");
        response.setDateHeader("Expires", 0);
        response.setContentType("image/jpeg");                    //设置响应正文的 MIME 类型为图片
        int width=60, height=20;
        /**创建一个位于缓存中的图像，宽度为 60，高度为 20 */
        BufferedImage image=new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        Graphics g = image.getGraphics();                         //获取用于处理图形上下文的对象，相当于画笔
        Random random = new Random();                             //创建生成随机数的对象
        g.setColor(getRandomColor(200,250));                      //设置图像的背景色
        g.fillRect(0, 0, width, height);                          //画一个矩形，坐标（0,0），宽度为 60，高度为 20
        g.setFont(new Font("Times New Roman",Font.PLAIN,18));     //设定字体格式
        g.setColor(getRandomColor(160,200));
        for(int i=0;i<130;i++){                                   //产生 130 条随机干扰线
                int x = random.nextInt(width);
```

```
        int y = random.nextInt(height);
        int xl = random.nextInt(12);
        int yl = random.nextInt(12);
        g.drawLine(x,y,x+xl,y+yl);            //在图像的坐标（x,y）和坐标（x+x1,y+y1）之间画干扰线
    }
    String strCode="";
    for (int i=0;i<4;i++){
        String strNumber=String.valueOf(random.nextInt(10));
            strCode=strCode+strNumber;
        //设置字体的颜色
        g.setColor(new Color(15+random.nextInt(120),15+random.nextInt(120),15+random.nextInt(120)));
        g.drawString(strNumber,13*i+6,16);    //将验证码依次画到图像上，坐标（x=13*i+6,y=16）
    }
    request.getSession().setAttribute("Code",strCode);      //把验证码保存到 Session 中
    g.dispose();                                            //释放此图像的上下文以及它使用的所有系统资源
    ImageIO.write(image, "JPEG", response.getOutputStream()); //输出 JPEG 格式的图像
    response.getOutputStream().flush();                     //刷新输出流
    response.getOutputStream().close();                     //关闭输出流
}
```

（2）在 ValidateCodeServlet 中添加一个获取随机颜色的方法，关键代码如下：

```
public  Color getRandomColor(int fc,int bc){
    Random random = new Random();
    Color randomColor = null;
    if(fc>255) fc=255;
    if(bc>255) bc=255;
    //设置 0～255 之间的随机颜色值
    int r=fc+random.nextInt(bc-fc);
    int g=fc+random.nextInt(bc-fc);
    int b=fc+random.nextInt(bc-fc);
    randomColor = new Color(r,g,b);
    return randomColor;                       //返回具有指定红色、绿色和蓝色值的不透明的 sRGB 颜色
}
```

（3）新建用户注册表单页 index.jsp，在该页的表单中使用<img>标签的 src 属性来调用 ValidateCodeServlet 类生成验证码，关键代码如下：

```
<form action="" method="post">
    <table align="center">
        <tr>
            <td>用户名：</td><td><input type="text" name="name" /></td>
        </tr>
        ……                                    //省略了其他注册信息的表单元素
        <tr>
            <td>验证码：</td><td><img alt="" src="validatecode" ></td>
        </tr>
        <tr>
            <td>输入验证码：</td><td><input type="text" name="code"/></td>
        </tr>
        <tr>
            <td colspan="2" align="center">
                <input type="submit" value="注  册" /><input type="reset" value="重  置" />
            </td>
        </tr>
    </table>
</form>
```

秘笈心法

由于服务器端的 Servlet 生成的图像首先是存放在缓存中的，为了使客户端获取到最新的图像，避免客户端通过缓存获取图像，应该在 Servlet 中通过设置特定 HTTP 响应头来禁止客户端缓存页面。在 Servlet 中，禁止客户端缓存页面的具体代码如下：

```
response.setHeader("Pragma", "No-cache");
response.setHeader("Cache-Control", "No-cache");
response.setDateHeader("Expires", 0);
```

实例 175　避免客户端访问的并发问题

光盘位置：光盘\MR\06\175

高级　实用指数：★★★★☆

实例说明

在 Web 应用程序开发或者网站开发中，一个 Web 应用可能会存在多个客户同时访问的情况，甚至可能同时访问同一个 Servlet，如果程序没有及时地处理并发问题，可能会返回给客户端错误的信息。本实例将讲解如何避免客户端的并发问题。运行本实例，如图 6.12 所示，同时打开两个浏览器，在表单中输入同样的数字，单击“等于”按钮提交之后，其中一个提交之后返回的值出现错误。

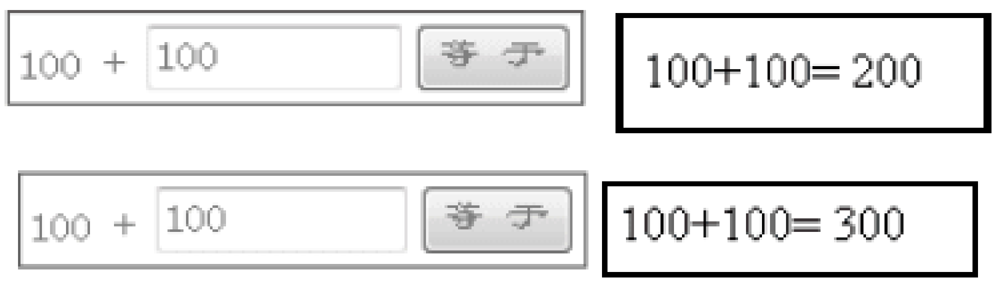


图 6.12　客户端并发访问出现的错误

关键技术

避免客户端并发的问题主要有以下两个解决办法：

（1）合理决定在 Servlet 中定义的变量的作用域。确定变量是全局变量还是局部变量，如果定义错误，客户端并发访问时可能会出现问题。

（2）多个线程同时访问共享数据而导致并发问题时，应该使用 Java 同步机制对多线程进行同步。Java 同步机制确保在任意时刻，只允许有一个线程执行共享数据的代码块，只有当这个线程退出同步代码块时，其他线程才允许执行同步代码块，因此可以避免并发所带来的问题。

设计过程

（1）新建名为 SynchronizationServlet 的 Servlet 类，在该类中定义一个全局变量，在 doPost()方法中使用 synchronized 关键字避免并发问题，关键代码如下：

```
private int    sum =100;//定义一个全局变量
public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        String num2 = request.getParameter("num2");
        int n2 = Integer.parseInt(num2);
        response.setContentType("text/html");
        response.setCharacterEncoding("UTF-8");
        PrintWriter out = response.getWriter();
        out.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\">");
        out.println("<HTML>");
        out.println("   <HEAD><TITLE>A Servlet</TITLE></HEAD>");
        out.println("   <BODY>");
        synchronized(this){                //同步以下代码块，同步之后，某一时刻只有当前线程可以访问
            out.println(sum+"+"+n2+"=");
            try{
                Thread.sleep(3000); //使当前线程休眠 3000 毫秒，然后继续执行
            }catch(Exception ex){
                ex.printStackTrace();
            }
            sum += n2;
            out.println(sum);
        }
        out.println("   </BODY>");
        out.println("</HTML>");
        out.flush();
        out.close();
}
```

（2）在 web.xml 中配置 SynchronizationServlet 类，关键代码如下：

```
<servlet>
    <servlet-name>SynchronizationServlet</servlet-name>
```

```
        <servlet-class>com.lh.servlet.SynchronizationServlet</servlet-class>
</servlet>
<servlet-mapping>
        <servlet-name>SynchronizationServlet</servlet-name>
        <url-pattern>/synchronization</url-pattern>
</servlet-mapping>
```

秘笈心法

在实际的开发应用过程中，应该考虑到客户端并发访问所带来的问题，解决并发问题的办法主要是合理决定在 Servlet 中定义的变量的作用域以及使用 Java 同步机制对线程进行同步。

实例 176　在 Servlet 中使用 JDBC 访问数据库

光盘位置：光盘\MR\06\176

初级

实用指数：★★★☆

实例说明

JDBC 是 Java API 中对数据库操作的重要组件，也是 Java 程序访问数据库的标准接口。为了帮助程序员编写可以在不同数据库引擎之间通用的代码，在 Java API 中提供了访问数据库的 JDBC API，使用 JDBC 中提供的接口和类可以在任何关系数据库中以相同的方式执行 SQL 语句。本实例将介绍如何在 Servlet 中通过 JDBC 技术来访问数据库。运行本实例，如图 6.13 所示，在表单页中输入用户注册信息，单击“注册”按钮后会将信息保存到数据库中。





id	name	pwd	sex	age	email
1	test	123	男	123	test@123.com
2	张三	123456	男	26	zhangsan@163.com

图 6.13　将用户注册信息保存到数据库中

关键技术

java.sql 包中提供了用于连接数据库的 JDBC API，使用 JDBC 连接不同的数据库需要加载不同数据库厂商所提供的数据库驱动类，具体的驱动类包可以到数据库厂商的官方网站进行下载。本实例使用的是 MySQL 数据库，该数据库的驱动包文件可以到 MySQL 官方网站进行下载。使用 JDBC 连接数据库的步骤如下。

（1）注册加载数据库连接的驱动程序。加载数据库驱动程序使用的是 Class.forName()方法，调用此方法会将指定的类加载到 JVM 中，关键代码如下：

```
Class.forName("com.mysql.jdbc.Driver");
```

（2）设置访问数据库连接 URL，关键代码如下：

```
String url = "jdbc:mysql://localhost:3306/db_database06";//连接 URL
```

（3）建立数据库连接。通过 JDBC API 中 DriverManager 类的 getConnection()方法来创建与数据库之间的连接，getConnection()方法需要接收 3 个 String 类型的参数，分别是连接 URL、用户名以及密码，关键代码如下：

```
Connection    con = DriverManager.getConnection(url, user, pwd);
```

（4）建立连接之后，使用数据库连接对象 Connection 创建用户操作 SQL 语句的 Statement 对象，关键代码如下：

```
Statement stmt = con.createStatement();
```

也可以用 Connection 对象创建 PreparedStatement 对象来执行 SQL 语句，使用的是 prepareStatement()方法，关键代码如下：

```
PreparedStatement pstmt = con.prepareStatement("select * from tb_user");
```

（5）通过 Statement 对象的 execute()方法，编译执行 sql 语句，关键代码如下：

```
String sql="update from tb_user set age=30 where userId=1";
stmt.execute (sql);
```

（6）关闭数据库连接。数据库用完后要及时关闭与数据库之间的连接，释放与数据库关联的资源，关键代码如下：

```
con.close();   //关闭连接
```

设计过程

（1）新建用户注册表单页 index.jsp，关键代码如下：

```
<form action="login" method="post">
        <table align="center">
                <tr>
                        <td>用户名：</td><td><input type="text" name="name" /></td>
                </tr>
                <tr>
                        <td>密码：</td><td><input type="password" name="pwd" /></td>
                </tr>
……              //此处省略了表单中其他选项
                <tr>
                        <td colspan="2" align="center">
                                <input type="submit" value="注  册" /><input type="reset" value="重  置" />
                        </td>
                </tr>
        </table>
</form>
```

（2）新建名为 UserInfo 的 JavaBean 类，用于封装用户的注册信息，关键代码如下：

```
public class UserInfo {
 private String id;                                          //用户编号
 private String name;                                        //用户名
 private String pwd ;                                        //密码
 private String sex;                                         //性别
 private int age;                                            //年龄
 private String email;                                       //Email
 public UserInfo(){}
……   //此处省略了属性的 getXXX()和 setXXX()方法
}
```

（3）在 Servlet 中使用 JDBC 访问数据库。可以将数据库连接的代码单独写成一个类，这样在其他的功能模块中可以实现代码的重用。新建名为 MySQLDBCon 的类，该类主要包含一个建立数据库连接的方法，关键代码如下：

```
public class MySQLDBCon {
 private static Connection conn = null;
 public static Connection getConn(){
        try{
                Class.forName("com.mysql.jdbc.Driver");                 //加载数据库驱动类
                String user = "root";                                   //用户名
                String pwd = "111";                                     //密码
                String url = "jdbc:mysql://localhost:3306/db_database06";  //连接 URL
                conn = DriverManager.getConnection(url,user,pwd);       //创建数据库连接
        }catch(Exception ex){
                ex.printStackTrace();
        }
        return conn;
 }
}
```

（4）新建名为 LoginDao 的类，该类中包含一个保存用户注册信息到数据库的方法和一个获得本类实例的静态方法，关键代码如下：

```
public class LoginDao {
 private static LoginDao instance = null;
```

```
public static LoginDao getInstance(){
        if(instance == null){
                instance = new LoginDao();
        }
        return instance;
}
//保存用户注册信息
public boolean saveUser(UserInfo user){
        Connection conn = null;
        try{
                conn = MySQLDBCon.getConn();                                          //建立数据库连接
                String sql = "insert into tb_userinfo(name,pwd,sex,age,email) values(?,?,?,?,?)";    //insert SQL 语句
                PreparedStatement pstmt = conn.prepareStatement(sql);
                pstmt.setString(1, user.getName());                                   //为 SQL 语句第 1 个参数赋值
                pstmt.setString(2, user.getPwd());                                    //为 SQL 语句第 2 个参数赋值
                pstmt.setString(3, user.getSex());                                    //为 SQL 语句第 3 个参数赋值
                pstmt.setInt(4, user.getAge());                                       //为 SQL 语句第 4 个参数赋值
                pstmt.setString(5, user.getEmail());                                  //为 SQL 语句第 5 个参数赋值
                pstmt.executeUpdate();                                                //执行 insert 语句
                return true;
        }catch(Exception ex){
                ex.printStackTrace();
        }finally{
                try {
                        conn.close();
                } catch (SQLException e) {
                        e.printStackTrace();
                }
        }
        return false;
}
}
```

（5）新建名为 LoginServlet 的 Servlet 类，在该类的 doPost()方法中获得表单提交的用户注册信息，并将这些信息封装到用户信息类 UserInfo 中，然后调用 LoginDao 中的方法将注册信息保存到数据库中，关键代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");                                        //设置请求字符编码格式
        /**以下是获得表单提交过来的值*/
        String name = request.getParameter("name");
        String pwd= request.getParameter("pwd");
        String sex = request.getParameter("sex");
        String age = request.getParameter("age");
        int userAge =0;
        if(age!=null&&!age.equals("")){
                userAge = Integer.parseInt(age);
        }
        String email = request.getParameter("email");
        /**以下代码将获得的表单值封装到用户信息对象中*/
        UserInfo user = new UserInfo();
        user.setName(name);
        user.setPwd(pwd);
        user.setSex(sex);
        user.setAge(userAge);
        user.setEmail(email);
        boolean res = LoginDao.getInstance().saveUser(user);                          //将用户注册信息保存到数据库
        ……//此处省略了一些非关键代码
}
```

■ 秘笈心法

在本实例中，将获取数据库连接的代码、表单数据以及数据库的操作都封装在单独的类中，而 Servlet 只是负责调用这些类，这样便充分地体现了 Servlet 作为控制器（Controller）角色的作用。

实例 177 利用 Servlet 访问数据库连接池

光盘位置：光盘\MR\06\177

高级

实用指数：★★★★

实例说明

由于建立数据库连接需要消耗大量的系统资源，频繁创建数据库连接会大大降低应用程序的性能，针对这一问题，可以使用数据库连接池，通过它可以提高访问数据库的效率。以实例 176 为基础，本实例将介绍如何利用 Servlet 访问数据库连接池。

关键技术

数据库连接池的工作原理是：首先会在连接池中建立一定数量的数据库连接，当程序需要访问数据库时，会从池中取出一个空闲连接，然后会将此连接锁定，标记为“忙”；当本次连接执行结束时，会将此连接放回连接池，并标记为“空闲”；当池中没有空闲连接时，池驱动程序会根据配置再创建指定数目的数据库连接。

可以在 Tomcat 服务器中配置数据库的连接池，由于 Tomcat 服务器的版本不同，具体的连接池配置也不相同，本实例是按照 Tomcat 7 来配置的，具体配置过程请参见设计过程。

设计过程

（1）在 Tomcat 服务器的安装文件目录中，包含一个 conf 文件夹，该文件夹中有一个名为 context.xml 的文件，主要是在该文件的<Context>元素中配置数据库的连接池，具体配置如下：

```
<Context   path="/mysql" reloadable="true" >
<!--
        name:               指定 Resource 的 JNDI 名字
        type:               指定 Resource 所属的 Java 类名
        auth:               指定管理 Resource 的 Manager，Container 表示由容器来管理 Resource
        maxActice:          处于活动状态的数据库连接的最大数目
        maxIdle:            处于空闲状态的数据库连接的最大数目
        maxWait:            指定数据库连接池中处于空闲状态连接的最长时间，以毫秒为单位
        username:           连接数据库的用户名
        password:           连接数据库的密码
        driverClassName:    连接数据库的 JDBC 驱动程序
        url:                连接数据库的 URL
 -->
<Resource name="jdbc/mysql" auth="Container"
            type="javax.sql.DataSource"
            maxActive="100"
            maxIdle="30"
            maxWait="10000"
            username="root"
            password="111"
            driverClassName="com.mysql.jdbc.Driver"
            url="jdbc:mysql://localhost:3306/db_database06"/>
    <ResourceLink global="jdbc/mysql" name="jdbc/mysql" type="javax.sql.DataSource"/>
    <!-- Default set of monitored resources -->
    <WatchedResource>WEB-INF/web.xml</WatchedResource>
</Context>
```

（2）配置完数据库连接池后，需要将 MySQL 数据库的驱动程序文件复制到 Tomcat 安装目录的 lib 文件夹中。

（3）建立数据库连接的 MySQLDBCon 类，关键代码如下：

```
public class MySQLDBCon {
 private static Connection conn = null;
 public static Connection getConn(){
        try{
               //通过这个类访问 Tomcat 配置文件 Context.xml 中指定的 JNDI 数据源
               InitialContext ctx = new InitialContext();
               //使用 lookup()方法查找匹配的 JNDI 名字，获得数据源
```

```
            DataSource ds = (DataSource)ctx.lookup("java:comp/env/jdbc/mysql");
            //从数据源中获得数据库连接
            conn = ds.getConnection();
        }catch(Exception ex){
            ex.printStackTrace();
        }
        return conn;
    }
}
```

秘笈心法

JNDI（Java Naming and Directory Interface），可以简单地把 JNDI 理解为一种将对象和名字绑定的技术，对象工厂负责生产出对象，这些对象都和唯一的 JNDI 名字绑定，外部程序通过 JNDI 名字来获取某个对象的引用。例如，在本实例中，在 Tomcat 容器中构造一个数据源，即 javax.sql.DataSource 的实例，然后把它发布为 JNDI 资源，名称为 jdbc/mysql，最后在 Java 应用中通过 JNDI API 中的 javax.nameing.Context 接口来获取这个数据源的引用。

实例 178　Servlet 实现的个人所得税计算器

光盘位置：光盘\MR\06\178

初级

实用指数：★★★★☆

实例说明

本实例将介绍如何通过 Servlet 来实现个人所得税的计算。运行本实例，如图 6.14 所示，在文本框中输入收入金额和起征金额，单击“计算个税”按钮，将调用 Servlet 计算个人所得税并显示结果。



图 6.14　个人所得税计算器

关键技术

工资、薪金的个人所得税计算公式为：（收入金额-起征点金额）×税率-速算扣除数。不同的工资范围其税率和速算扣除数也不同。薪资在扣除起征点金额之后的范围的税率以及速算扣除数如表 6.2 所示。

表 6.2　薪资在扣除起征点金额之后的范围的税率和速算扣除数

薪资在扣除起征点金额之后的范围	税　率	速算扣除数
金额在 0～500 之间	5%	0
金额在 501～2000 之间	10%	25
金额在 2001～5000 之间	15%	125
金额在 5001～20000 之间	20%	375
金额在 20001～40000 之间	25%	1375
金额在 40001～60000 之间	30%	3375
金额在 60001～80000 之间	35%	6375
金额在 80001～100000 之间	40%	10375
金额在 100001 以上	45%	15375

设计过程

（1）新建表单页 index.jsp，关键代码如下：

```
<form action="incometax" method="post">
		<table>
			<tr>
				<td>收入金额：</td>
				<td>
					<input type="text" name="laborage"   />元
				</td>
			</tr>
			<tr>
				<td>起征金额：</td>
				<td>
					<input type="text" name="startpoint" value="2000" />元
				</td>
			</tr>
			<tr>

				<td align="center" colspan="2">
					<input type="submit"   value="计算个税" />
				</td>
			</tr>
		</table>
</form>
```

（2）新建名为 IncomeTaxServlet 的 Servlet 类，该类中实现计算个人所得税的方法，在 doPost()方法中根据获得的收入金额和起征金额来计算个人所得税，关键代码如下：

```
//计算个人所得税
public double getTax(double charge){
	double tax = 0;
	if(charge<=0){
		tax = 0;
	}else if(charge>0&&charge<=500){
		tax = charge*0.05;
	}else if(charge>500&&charge<=2000){
		tax = charge*0.1-25;
	}else if(charge>2000&&charge<=5000){
		tax = charge*0.15-125;
	}else if(charge>5000&&charge<=20000){
		tax = charge*0.2-375;
	}else if(charge>20000&&charge<=40000){
		tax = charge*0.25-1375;
	}else if(charge>40000&&charge<=60000){
		tax = charge*0.30-3375;
	}else if(charge>60000&&charge<=80000){
		tax = charge*0.35-6375;
	}else if(charge>80000&&charge<=100000){
		tax = charge*0.4-10375;
	}else if(charge>100000){
		tax = charge*0.45-15375;
	}
	return tax;
}
public void doPost(HttpServletRequest request, HttpServletResponse response)
		throws ServletException, IOException {
	double laborage = Double.parseDouble(request.getParameter("laborage"));		//获得表单提交的工资收入
	double startPoint = Double.parseDouble(request.getParameter("startpoint"));	//获得表单提交的征税起点金额
	double   myTax = this.getTax(laborage - startPoint);					//调用计算个人所得税的方法
	request.setAttribute("Tax", myTax);							//将个人所得税的值保存在请求中
	request.getRequestDispatcher("result.jsp").forward(request, response);		//请求转发到 result.jsp 页
}
```

（3）在 web.xml 中配置 IncomeTaxServlet 类，关键代码如下：

```
<servlet>
    <servlet-name>IncomeTaxServlet</servlet-name>
    <servlet-class>com.lh.servlet.IncomeTaxServlet</servlet-class>
```

```
</servlet>
<servlet-mapping>
    <servlet-name>IncomeTaxServlet</servlet-name>
    <url-pattern>/incometax</url-pattern>
</servlet-mapping>
```

（4）新建 result.jsp 页，从请求范围内取出个人所得税的计算结果并显示，关键代码如下：

```
<table>
    tr>
        <td>您应交纳的个人所得税为：</td>
        <td>
            <%=request.getAttribute("Tax").toString() %>元
        </td>
    </tr>
</table>
```

秘笈心法

本实例实现的只是工资、薪金的个人所得税计算，而实际的个人所得税还包括很多种，例如，个体工商户的生产经营所得税、企业单位的承包经营所得税、劳动报酬所得税等，其具体的税率算法也有所差异。如果读者对此感兴趣，可以查阅相关资料进行了解，此处不再详细介绍。

实例 179　利用 Servlet 实现用户永久登录

光盘位置：光盘\MR\06\179

高级

实用指数：★★★★

实例说明

在访问一些网站时，用户在登录网站之后，网站会将该用户信息保存一段时间，当该用户再访问该网站时，不需要输入用户名和密码就会自动进入登录状态。运行本实例，如图 6.15 所示，输入账号和密码，然后选择有效期为“30 天内有效”，单击“登录”按钮之后，该账号会保存 30 天，再次访问时不必登录，会直接进入登录状态，只有单击“注销登录”超链接时，该用户的登录状态才会失效。

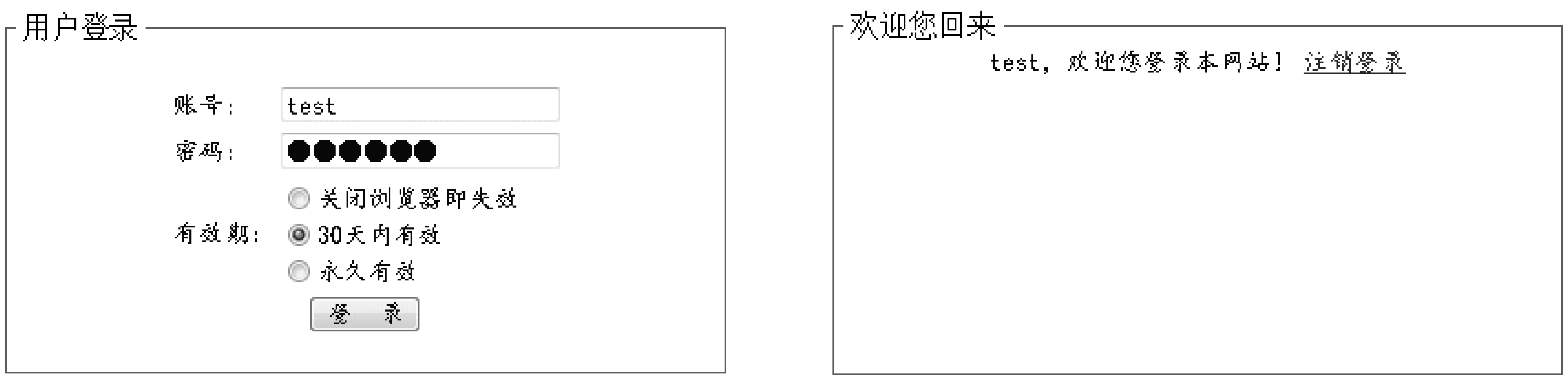


图 6.15　利用 Servlet 实现用户永久登录

关键技术

本实例主要是在 Servlet 中通过 Cookie 技术来实现的。首先在 Servlet 中获得用户输入的账号、密码和有效期，然后将账号信息保存在 Cookie 中，并设置该 Cookie 的最大保存时间，然后将此 Cookie 保存在客户端的 Cookie 中。

本实例的实现还用到了 MD5 加密技术。考虑到账号密码的安全性，由于不能将密码保存在 Cookie 中，因此可以在 Servlet 中，通过 MD5 加密算法将用户账号生成一个密钥并保存在 Cookie 中，然后在用户登录页中，就可以根据该密钥来判断页面显示的是用户登录还是登录后的状态。MD5 加密是通过 java.security.MessageDigest 类实现的，可以使用 MD5 或 SHA 字符串类型的值作为参数来构造一个 MessageDigest 类对象，并使用 update()方法更新该对象，最后通过 digest()方法完成加密运算，代码如下：

```
String pwd="123456";
MessageDigest md = MessageDigest.getInstance("MD5");    //创建具有指定算法名称的摘要
md.update(pwd.getBytes());                              //使用指定的字节数组更新摘要
byte mdBytes[] = md.digest();                           //进行哈希计算并返回一个字节数组
```

设计过程

（1）新建名为 MakeMD5 的类，该类实现了将字符串转换为 MD5 值的方法，关键代码如下：

```
public class MakeMD5 {
 public final static String getMD5(String str){
        char hexDiagitArr[] ={'0','1','2','3','4','5','6','7','8','9','a','b',
'c','d','e','f'};
        MessageDigest digest = null;
        try{
                digest= MessageDigest.getInstance("MD5");      //创建 MD5 算法摘要
                digest.update(str.getBytes());                 //更新摘要
                byte mdBytes[] = digest.digest();              //加密并返回字节数组
                //新建字符数组，长度为 myBytes 字节数组的 2 倍，用于保存加密后的值
                char newCArr[] = new char[mdBytes.length*2];
                int k=0;
                for(int i=0;i<mdBytes.length;i++){             //循环字节数组
                        byte byte0 = mdBytes[i];               //获得每一个字节
                        newCArr[k++] = hexDiagitArr[byte0 >>> 4 &0xf];
                        newCArr[k++] = hexDiagitArr[byte0 & 0xf];
                }
                return String.valueOf(newCArr);                //返回加密后的字符串
        }catch(Exception ex){
                ex.printStackTrace();
        }
        return null;
 }
}
```

（2）新建用户登录页 index.jsp，该页中包含一个用户登录表单和一个登录之后状态的显示信息，如果用户第一次访问该页会显示用户登录表单，并不会显示登录之后的信息，当用户登录之后再次访问用户登录页时，会判断 Servlet 返回的 Cookie 信息，根据 Cookie 信息来决定是否显示用户登录之后的信息，关键代码如下：

```
<body>
  <%
        if(loginFlag){
   %>
   <fieldset class="style1" ><legend>欢迎您回来</legend>
                <table align="center">
                        <tr>
                                <td><%=account %>，欢迎您登录本网站！</td>
                                <td align="center">
                                        <a href="<%=basePath%>foreverlogin?action=logout">注销登录</a>
                                </td>
                        </tr>
                </table>
   </fieldset>
   <%}else{ %>
  <fieldset class="style1"><legend>用户登录</legend>
     <form action="foreverlogin?action=login" method="post">
        <table align="center">
                <tr>
                        <td>账号：</td>
                        <td><input type="text" name="account"></td>
                </tr>
                <tr>
                        <td>密码：</td>
                        <td><input type="password" name="pwd"></td>
                </tr>
                <tr>
                        <td>有效期：</td>
                        <td>
                                <input type="radio" name="timeout" value="-1" checked="checked">
                                关闭浏览器即失效<br/>
                                <input type="radio" name="timeout" value="<%=30*24*60*60 %>">
                                30 天内有效<br/>
                                <input type="radio" name="timeout" value="<%=Integer.MAX_VALUE %>">
                                永久有效
```

```
                </td>
            </tr>
            <tr>
                <td colspan="2" align="center"><input type="submit" value="登　录" ></td>
            </tr>
        </table>
    </form>
  </fieldset>
  <%} %>
</body>
```

（3）新建名为 ForeverLoginServlet 的 Servlet 类，在该类的 doPost()方法中根据提交过来的 action 参数值来判断调用用户登录方法或用户注销的方法，关键代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");                    //设置请求编码格式
        response.setCharacterEncoding("UTF-8");                   //设置响应编码格式
        String action = request.getParameter("action");          //获得 action 参数，主要判断是登录还是注销
        if("login".equals(action)){
            this.login(request, response);                        //调用 login()方法
        }else if("logout".equals(action)){
            this.logout(request, response);                       //调用 logout()方法
        }
}
/**
 * 该方法处理用户登录
 */
public void login(HttpServletRequest request,HttpServletResponse response)
            throws ServletException, IOException{
        String account = request.getParameter("account");         //获得账号
        String pwd = request.getParameter("pwd");                 //获得密码
        int timeout= Integer.parseInt(request.getParameter("timeout"));  //获得登录保存期限
        String md5Account = MakeMD5.getMD5(account);              //将账号加密
        account = URLEncoder.encode(account,"UTF-8");             //如果账号是中文，需要转换 Unicode 才能保存在 Cookie 中
        Cookie accountCookie = new Cookie("account",account);     //将账号保存在 Cookie 中
        accountCookie.setMaxAge(timeout);                         //设置账号 Cookie 的最大保存时间
        Cookie md5AccountCookie = new Cookie("md5Account",md5Account);//将加密后的账号保存在 Cookie 中
        md5AccountCookie.setMaxAge(timeout);                      //设置加密后的账号最大保存时间
        response.addCookie(accountCookie);                        //写到客户端的 Cookie 中
        response.addCookie(md5AccountCookie);                     //写到客户端的 Cookie 中
        try {
            Thread.sleep(1000);                                   //将此线程暂停 1 秒后继续执行
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        response.sendRedirect("index.jsp?"+System.currentTimeMillis());  //将页面重定向到用户登录页
}
/**
 * 该方法处理用户注销
 */
public void logout(HttpServletRequest request,HttpServletResponse response)
            throws ServletException, IOException{
        Cookie accountCookie = new Cookie("account","");          //创建一个空的 Cookie
        accountCookie.setMaxAge(0);                               //设置此 Cookie 保存时间为 0
        Cookie md5AccountCookie = new Cookie("md5Account","");    //创建一个空的 Cookie
        md5AccountCookie.setMaxAge(0);                            //设置此 Cookie 保存时间为 0
        response.addCookie(accountCookie);                        //写到客户端 Cookie 中，将覆盖名为 account 的 Cookie
        response.addCookie(md5AccountCookie);                     //写到客户端 Cookie 中，将覆盖名为 md5AccountCookie 的 Cookie 值
        try {
            Thread.sleep(1000);                                   //将此线程暂停 1 秒后继续执行
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        //将页面重定向到用户登录页
        response.sendRedirect("index.jsp?"+System.currentTimeMillis());
}
```

（4）在 index.jsp 页中，设置一个保存是否登录的标记 loginFlag 为 false，通过 request 内置对象获得所有 Cookie 信息的数组，循环该数组查找账号的 Cookie 信息和加密账号之后的 Cookie 信息，然后通过 MD5 算法将账号加密生成密钥，将该密钥值与 Cookie 中保存的加密账号比较，如果两值匹配则将 loginFlag 标记改为 true，然后在页面中根据 loginFlag 的值来判断显示用户登录之后的信息，关键代码如下：

```
<%
boolean loginFlag = false;                                    //设置一个变量，用于保存是否登录
String account = null ;                                       //声明用于保存从 Cookie 中读取的账号
String md5Account = null;                                     //声明用于保存从 Cookie 中读取的加密的账号
Cookie cookieArr[] = request.getCookies();                    //获取请求中所有的 Cookie
if(cookieArr!=null&&cookieArr.length>0){
        for(Cookie cookie : cookieArr){                       //循环 Cookie 数组
                if(cookie.getName().equals("account")){
                        account = cookie.getValue();          //找到账号的 Cookie 值
                        account = URLDecoder.decode(account,"UTF-8");   //解码，还原中文字符串的值
                }
                if(cookie.getName().equals("md5Account")){
                        md5Account = cookie.getValue();       //找到加密账号的 Cookie 值
                }
        }
}
if(account!=null&&md5Account!=null){
        loginFlag = md5Account.equals(MakeMD5.getMD5(account));
}
%>
```

秘笈心法

当 Servlet 向客户端写 Cookie 时，关键是要通过 Cookie 类的 setMaxAge(int expiry)方法来设置 Cookie 的有效期。参数 expiry 以秒为单位，如果 expiry 大于零，就指示浏览器在客户端硬盘上保持 Cookie 的时间为 expriy 秒；如果 expiry 等于零，就指示浏览器删除当前 Cookie；如果 expiry 小于零，就指示浏览器不要把 Cookie 保存到客户端硬盘，当浏览器进程关闭时，Cookie 也就消失。

第 7 章

过滤器与监听器技术

- Servlet 过滤器
- 监听器的应用

7.1　Servlet 过滤器

Servlet 过滤器从表面的字意理解为经过一层层的过滤处理才达到使用的要求，而其实 Servlet 过滤器就是服务器与客户端请求与响应的中间层组件，在实际项目开发中 Servlet 过滤器主要用于对浏览器的请求进行过滤处理，将过滤后的请求再转给下一个资源。其实 Servlet 过滤器与 Servlet 十分相似，只是多了一个具有拦截浏览器请求的功能。过滤器可以改变请求的内容来满足客户的需求，对开发人员来说，这点在 Web 开发中具有十分重要的作用。

实例 180	创建过滤器 光盘位置：光盘\MR\07\180	初级 实用指数：★★

实例说明

本实例将介绍如何创建一个过滤器，并使用过滤器在打开页面的同时输出信息，此功能是由过滤器处理完成的，运行结果如图 7.1 所示。

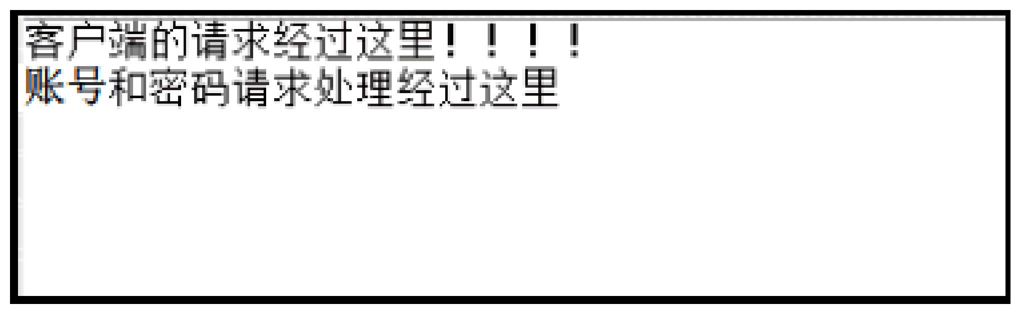


图 7.1　创建过滤器

关键技术

Servlet 过滤器实现了 Filter 接口，在该接口中定义了以下几个方法。

- ❑ init()：程序启动时调用此方法，用于初始化该 Filter。
- ❑ doFilter()：客户请求服务器时会经过这里，是具体执行过滤器代码。
- ❑ destroy()：程序关闭时调用此方法，用于销毁一些资源。

以上 3 个方法反映了 Filter 的生命周期，其中 init()和 destroy()方法只会被调用一次，分别在 Web 程序加载和卸载时调用，而 doFilter()方法每次有客户端请求就会被调用一次。

设计过程

（1）创建过滤器类 FirstFilter，主要代码如下：

```
package com.mr;
public class FirstFilter implements Filter {
 private FilterConfig filterConfig;
 //初始化方法
 public void init(FilterConfig filterConfig) throws ServletException {
         this.filterConfig = filterConfig;
 }
 //具体执行的方法
 public void doFilter(ServletRequest request, ServletResponse response,FilterChain filterChain) throws IOException{
         try {
                 System.out.println("客户端的请求经过这里！！！！ ");
                 filterChain.doFilter (request, response);
                 System.out.println("账号和密码请求处理经过这里");
         } catch (ServletException e) {
                 System.out.println("客户端请求失败");
         }catch(IOException io){
                 System.out.println("账号和密码请求失败");
         }
 }
 //销毁过滤器
 public void destroy() {
```

```
        this.filterConfig=null;
    }
}
```

（2）在 web.xml 中配置过滤器，关键代码如下：

```
<filter>
        <filter-name>FirstFilter</filter-name><!--过滤器名称 -->
        <filter-class>com.mr.FirstFilter</filter-class><!--过滤器的实现类 -->
</filter>
<filter-mapping>
       <filter-name>firstFilter</filter-name>     <!--映射过滤器名称 -->
        <url-pattern>/*</url-pattern>               <!--使用通配符*什么请求都经过滤器 -->
</filter-mapping>
```

秘笈心法

过滤器在 web.xml 配置时，<filter>元素用来定义一个过滤器，<filter-mapping>元素用于为过滤器映射特定的 URL。注意在配置多个 Filter 时执行有先有后，规定是<filter-mapping>配置在前面的 Filter 执行要早于配置在后面的 Filter，另外要注意多个 Filter 可能会互相影响。

实例 181　防盗链过滤器　　初级

光盘位置：光盘\MR\07\181　　实用指数：★★

实例说明

本实例将介绍如何使用过滤器技术，防止通过其他 URL 地址直接访问本站资源。运行本实例，当 URL 地址不是本站地址时，在网页中将显示错误提示信息，实例运行效果如图 7.2 所示。

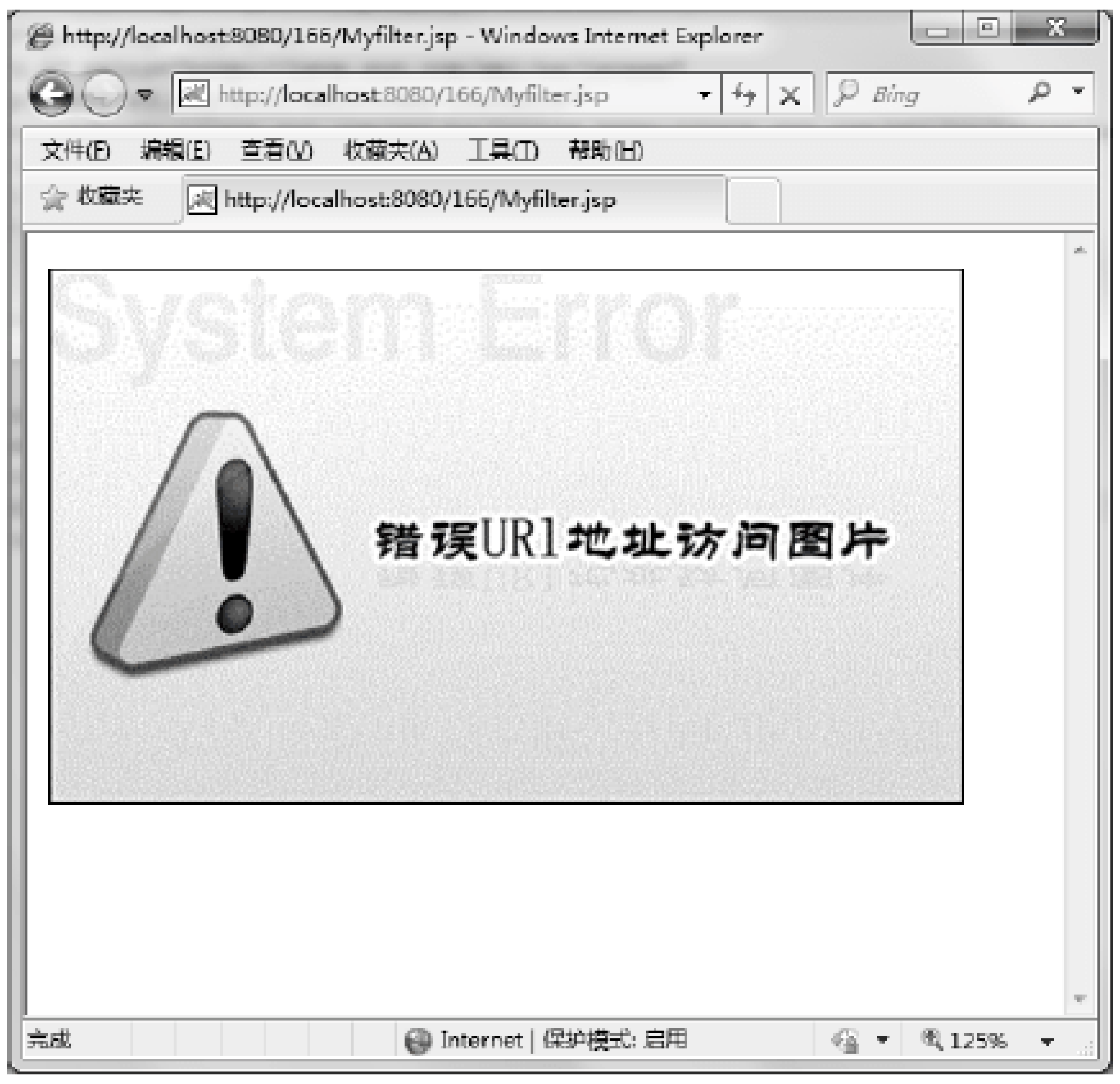


图 7.2　防盗链过滤器

关键技术

本实例主要应用 request 对象的 getHeader()方法获取信息头来源地址，若是来自其他网站就弹出错误图片。getHeader()方法的语法结构如下：

```
public String getHeader(String headerName)
```

参数说明

headerName：指定字符串类型的响应头名称。

设计过程

（1）创建 Filter 过滤器的实现类 ImageFilter，在 doFilter()方法中对 request 进行验证，实现将图片显示在页

面之前，验证客户端请求是否来自本网站，主要代码如下：

```
public class ImageFilter implements Filter {
public void init(FilterConfig config) throws ServletException {
}
public void doFilter(ServletRequest req, ServletResponse res,
            FilterChain chain) throws IOException, ServletException {
    HttpServletRequest request = (HttpServletRequest) req;              //request 对象
    HttpServletResponse response = (HttpServletResponse) res;           //response 对象
    String imurl = request.getHeader("imurl");                          //链接的来源地址
    if (imurl == null || !imurl.contains(request.getServerName())) {    //判断访问来源
        request.getRequestDispatcher("/errorimage.gif").forward(request,  //显示错误图片
                response);
    } else {
        chain.doFilter(request, response);                              //正常显示图片
    }
}
public void destroy() {
}
}
```

（2）在 web.xml 中配置 Filter，该过滤器是从 request 信息头中获取请求来源，主要代码如下：

```
<filter>
<filter-name>imageFilter</filter-name>
<filter-class>com.mr.filter.ImageFilter</filter-class>
</filter>
<filter-mapping>
<filter-name>imageFilter</filter-name>
<url-pattern>/images/*</url-pattern>
</filter-mapping>
```

秘笈心法

本实例的 Filter 是对 images 下所有的资源有效，如果是“/*”就是对工程包下所有的资源有效。

实例 182 日志记录过滤器

光盘位置：光盘\MR\07\182

初级

实用指数：★★

实例说明

在实际的项目开发过程中，经常需要在项目运行时记录并在控制台中输出运行时的日志信息，便于查看项目的运行状况。本实例将介绍如何应用过滤器实现日志记录。运行本实例，将在控制台中输出项目运行时的日志信息，如图 7.3 所示。

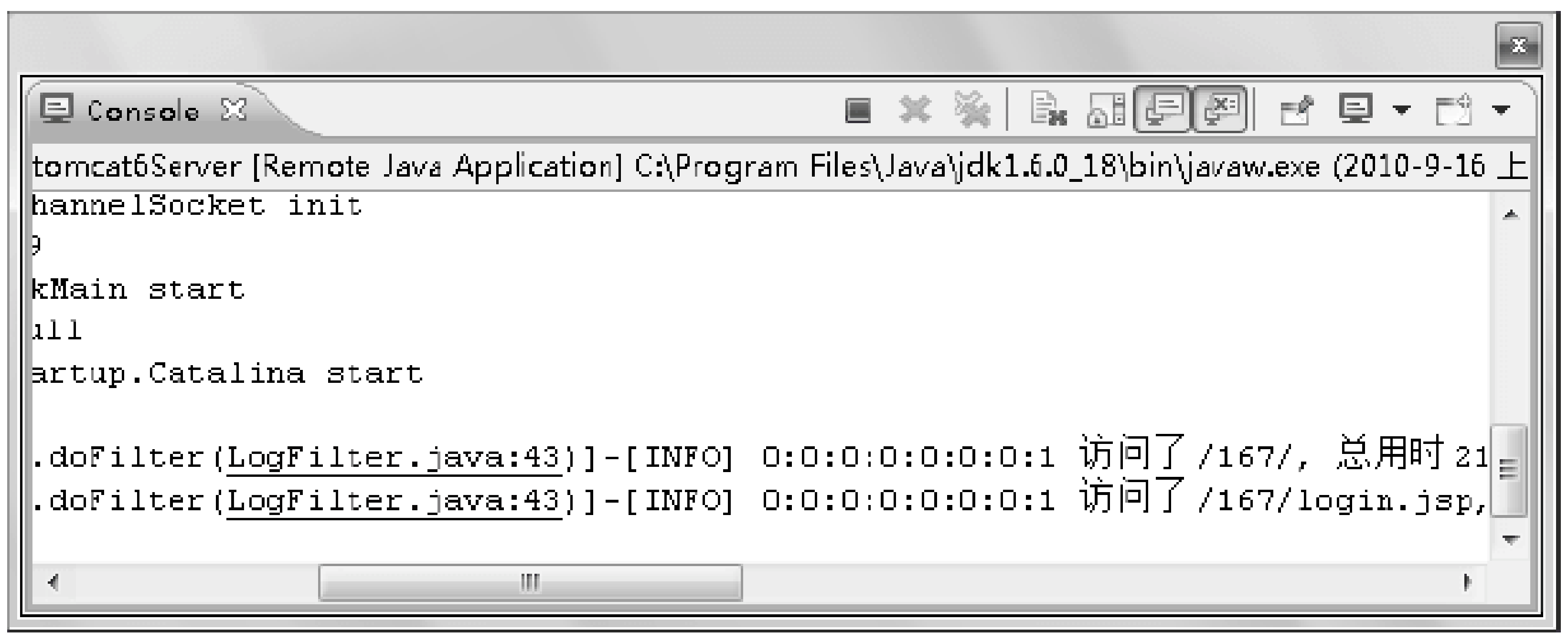


图 7.3　控制台输出日志信息

关键技术

本实例主要应用 Apache 的 Log4j 组件输出日志信息。该组件主要用于日志管理。Logger 是 Log4j 的日志记

录器，它是 Log4j 的核心组件。

在程序中可以使用 Logger 类的不同方法来输出各种级别的日志信息，Log4j 会根据配置的当前日志级别决定输出哪些日志。对应各种级别日志的输出方法如下：

（1）DEBUG 日志可以使用 Logger 类的 debug()方法输出日志消息，语法格式如下：

```
logger.debug(Object message)
```

message：输出的日志消息，如“logger.debug("调试日志")”。

（2）INFO 日志可以使用 Logger 类的 info()方法输出日志消息，语法格式如下：

```
logger.info(Object message)
```

message：输出的日志消息，如“logger.info("消息日志")”。

（3）WARN 日志可以使用 Logger 类的 warn()方法输出日志消息，语法格式如下：

```
logger.warn(Object message)
```

message：输出的日志消息，如“logger.warn("警告日志")”。

（4）ERROR 日志可以使用 Logger 类的 error()方法输出日志消息，语法格式如下：

```
logger.error(Object message)
```

message：输出的日志消息，如“logger.error("数据库连接失败")”。

（5）FATAL 日志可以使用 Logger 类的 fatal()方法输出日志消息，语法格式如下：

```
logger.fatal(Object message)
```

message：输出的日志消息，如“logger.fatal("内存不足")”。

设计过程

（1）创建日志 Filter 实现类 LogFilter.java，主要是在初次调用时开始记录，执行时获取访问的 URI 和执行前的时间，关键代码如下：

```
public class LogFilter implements Filter {
private Log log = LogFactory.getLog(this.getClass());
private String filterName;
public void init(FilterConfig config) throws ServletException {
        filterName = config.getFilterName();                        //获取 Filter 的 name
        log.info("启动  Filter: " + filterName);                     //启动 Filter
}
public void doFilter(ServletRequest req, ServletResponse res,FilterChain chain)
                throws IOException, ServletException {
        HttpServletRequest request = (HttpServletRequest) req;
        HttpServletResponse response = (HttpServletResponse) res;
        long startTime = System.currentTimeMillis();                //运行前的时间
        String requestURI = request.getRequestURI();                //获取访问的 URI
        requestURI = request.getQueryString() == null ? requestURI  //所有的地址栏参数对比
                        : (requestURI + "?" + request.getQueryString());
        chain.doFilter(request, response);
        long endTime = System.currentTimeMillis();
        //消耗的总时间
        log.info(request.getRemoteAddr() + "  访问了  " + requestURI + ", 总用时  "+ (endTime - startTime) + "  毫秒。");
}
public void destroy() { //销毁时记录日志
}
}
```

（2）使用日志记录需要 commons-logging 的 Log4j 来输出日志，本实例输出格式如下：

```
log4j.rootLogger=INFO, A1
log4j.appender.A1=org.apache.log4j.ConsoleAppender
log4j.appender.A1.layout=org.apache.log4j.PatternLayout
log4j.appender.A1.layout.ConversionPattern=%-d{yyyy-MM-dd HH\:mm\:ss,SSS} [%l]-[%p] %m%n
```

秘笈心法

Filter 日志最大的优点在于可拆卸性，当不需要记录日志功能时，只需要将 Filter 配置注释掉即可。

实例 183 字符替换过滤器

光盘位置：光盘\MR\07\183

初级

实用指数：★★★

实例说明

有时对网站内容进行控制是用来防止输出非法内容或者敏感内容的，常规的办法是保存数据库之前对非法内容进行替换，但这种办法具有局限性，工作量大并且耦合性比较高。本实例将介绍如何使用过滤器，对非法字符和关键字进行过滤处理，实例运行结果如图 7.4 所示。

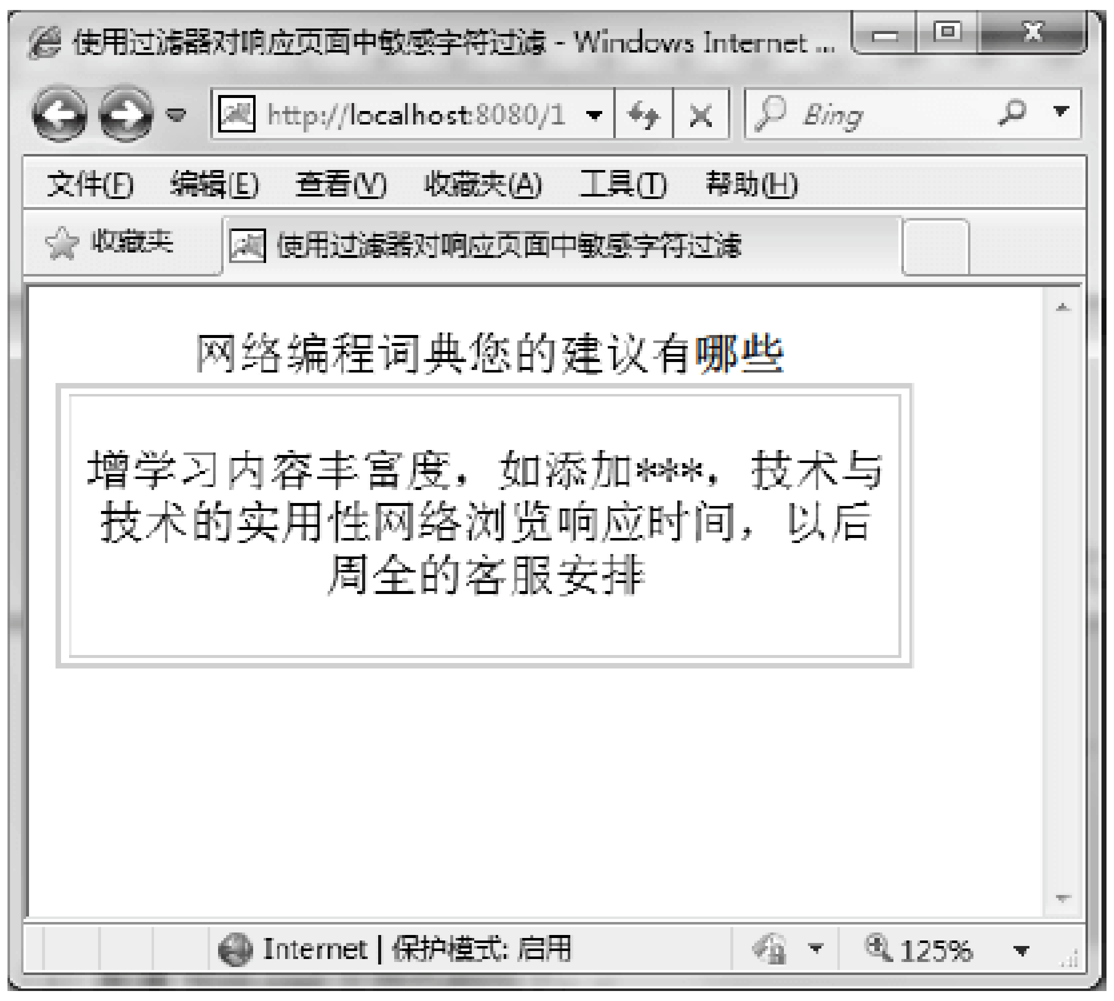


图 7.4 运行结果

关键技术

实现本实例，可以在服务器端使用过滤器技术。在 Web 服务器获得用户的请求之前，过滤器可以访问该请求。在 Web 服务器将输出响应发送给用户之前，过滤器还可以访问该响应。所以在过滤器中访问该响应信息，然后将该响应转换为自定义的响应，最后将过滤后的自定义响应内容返回给客户端。

在 java.servlet.http 包中，包含了一个名为 HttpServletResponseWrapper 的类，该类的对象表示一个自定义的响应对象，它实现了 HttpServletResponse 接口，其构造方法通过传入的 HttpServletResponse 类型的参数，将响应转换为自定义的响应。构造方法的语法结构如下：

```
public HttpServletResponseWrapper(HttpServletResponse response)
```

设计过程

（1）创建 Cr.java 类文件，作用是处理对页面响应的内容，用 toString()方法进行重载，然后将页面中的内容转换成字符串，关键代码如下：

```
public class Cr extends HttpServletResponseWrapper {
    private CharArrayWriter output;
    public String toString() {
       return output.toString();
    }
    public Cr(HttpServletResponse response){
          super(response);
          this.output=new CharArrayWriter();
    }
    public PrintWriter getWriter(){
          return new PrintWriter(output);
    }
}
```

（2）创建过滤器实现类 CtFilter.java，在 doFilter()方法中获取页面的响应，然后对这个响应内容进行处理并生成自定义的响应，把敏感字去掉并替换成“***”，再返回给客户端，关键代码如下：

```
public class CtFilter extends HttpServlet implements Filter {
      public void init(FilterConfig filterConfig) throws ServletException {
      }
      public void doFilter(ServletRequest request, ServletResponse response,FilterChain filterChain)
                  throws ServletException,IOException {
         response.setCharacterEncoding("gb2312");
         PrintWriter out = response.getWriter();
         Cr wrapper = new Cr((HttpServletResponse)response);
         filterChain.doFilter(request, wrapper);
         String resStr = wrapper.toString().trim();
         String newStr = "";
         if (resStr.indexOf("混蛋") > 0) {
                  newStr = resStr.replace("混蛋","***");
         }
         out.println(newStr);
      }
}
```

（3）在 web.xml 文件中配置过滤器，关键代码如下：

```
<filter>
         <filter-name>cr</filter-name>
         <filter-class>com.mr.CtFilter</filter-class>
</filter>
<filter-mapping>
 <filter-name>cr</filter-name>
 <url-pattern>/*</url-pattern>
</filter-mapping>
```

秘笈心法

如果响应 response 输出的内容为字符类内容，则调用 getWriter()方法；如果为二进制内容或图像数据等，就需要调用 getOutputStream()方法，本实例只覆盖了 getWriter()方法。

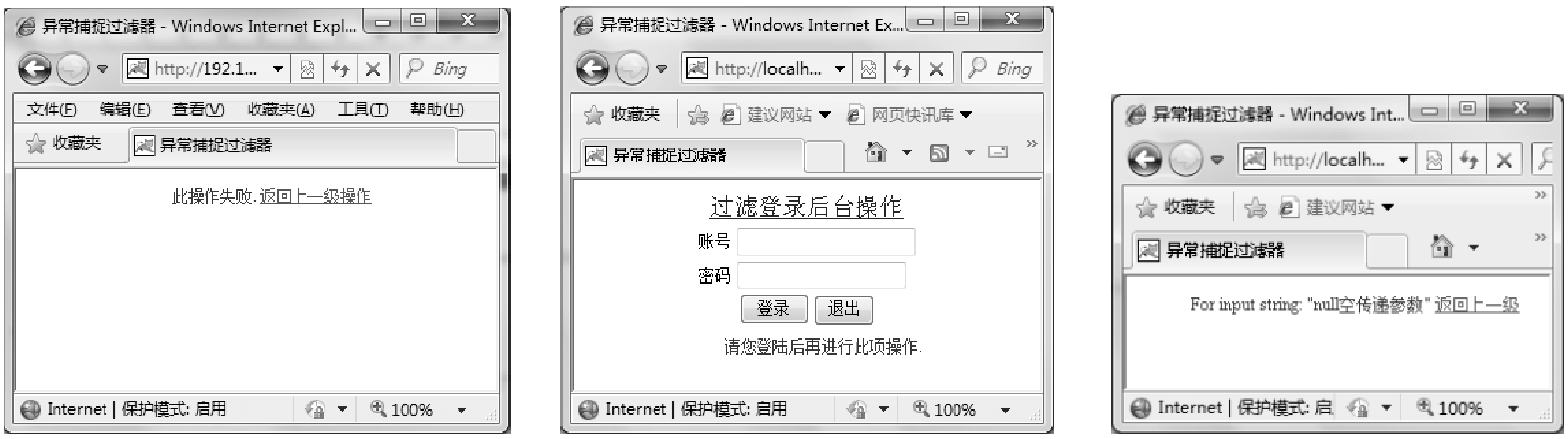
实例 184　异常捕获过滤器

光盘位置：光盘\MR\07\184

初级　实用指数：★★

实例说明

本实例将介绍如何实现异常捕获过滤器。运行本实例，单击异常超链接，会弹出相应的处理页面，运行结果如图 7.5 所示。



（a）　　（b）　　（c）

图 7.5　运行结果

关键技术

本实例主要是在过滤器 Filter 的 doFilter()方法中对执行过滤器链的 chain 的 doFilter()语句处添加 try…catch 异常捕获语句，然后在 catch 语句中循环异常对象，直到找出根异常为止。

设计过程

（1）创建 Filter 实现类 ExceptionFilter.java，利用 throwable 抛出异常去捕捉异常原因并转到相应的页面中，主要代码如下：

```
public class ExceptionFilter implements Filter {
public void destroy() {
}
public void doFilter(ServletRequest request, ServletResponse response,FilterChain chain) throws IOException, ServletException {
        try {
                chain.doFilter(request, response);
        } catch (Exception e) {                                                              //如果有异常则捕捉
                Throwable rootCause = e;
                while (rootCause.getCause() != null) {
                        rootCause = rootCause.getCause();
                }
                String errormessage = rootCause.getMessage();                              //返回此异常的详细消息字符串
                errormessage = errormessage == null ? "异常：" + rootCause.getClass().getName()
                                : errormessage;                                           //中止传递异常的原因
                request.setAttribute("errormessage", errormessage);
                request.setAttribute("e", e);
                if (rootCause instanceof LoginException) {                                  //转到登录页面
                        request.getRequestDispatcher("/LoginException.jsp").forward(request, response);
                } else if (rootCause instanceof OperationException) {
                                                                                          //转到操作页面
                        request.getRequestDispatcher("/OperationException.jsp").forward(request, response);
                } else {
                        request.getRequestDispatcher("/exception.jsp").forward(request,      //其他异常
                                        response);
                }
        }
}
public void init(FilterConfig arg0) throws ServletException {
}
}
```

（2）创建 LoginException.jsp 登录异常页面，主要代码如下：

```
<div align="center" style="font-size: large;">后台操作</div>
<form action="">
<table align="center">
        <tr>
                <td>账号</td>
                <td><input type="text" name="account" /></td>
        </tr>
        <tr>
                <td>密码</td>
                <td><input type="password" name="password" /></td>
        </tr>
        <tr>
                <td align="center" colspan="2"><input type="submit" value=" 登录 " />
                <input type="submit" value="退出"/></td>
        </tr>
</table>
</form>
<div class="error" align="center">
${ errormessage }
</div>
```

（3）创建 OperationException 操作异常页面，主要代码如下：

```
<div class="error" align="center">
${ errormessage } <a href="javascript:history.go(-1); ">返回上一级操作</a>
</div>
```

（4）创建 Exceptionmain.jsp 页面，关键代码如下：

```
<%
String action = request.getParameter("action");
if("OperationException".equals(action)){
        throw new OperationException("此操作失败.  ");
```

```
}
else if("LoginException".equals(action)){
        throw new LoginException("请您登录后再进行此项操作.  ");
}
else if("exception".equals(action)){
        Integer.parseInt("null 空传递参数");
}
%>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>异常捕捉过滤器</title>
</head>
<body>
<table align="left" cellpadding="2" cellspacing="2">
<Tr><td><a href="${ pageContext.request.requestURI }?action=OperationException">过滤操作</a></td></Tr>
<tr><td><a href="${ pageContext.request.requestURI }?action=LoginException">过滤登录</a></td></tr>
<Tr><td><a href="${ pageContext.request.requestURI }?action=exception">过滤异常</a></td></Tr>
</table>
```

秘笈心法

Throwable 类是 Java 语言中所有错误或异常的超类，只有当对象是此类的实例时，才能通过 Java 虚拟机或者 Java throw 语句抛出，类似的只有此类或其子类之一才可以是 catch 子句中的参数类型。

实例 185	验证用户身份 Filter 过滤器 光盘位置：光盘\MR\07\185	初级 实用指数：★★

实例说明

在进行用户的首次身份认证后都会在 session 中留下相应的用户对象作为标识，在以后的操作中，只需要在进行身份验证的页面或 Servlet 中查看相应的 session 即可。最有效的验证方法就是通过过滤器对一批页面或 Servlet 统一进行身份验证，这样在设计页面或 Servlet 时就不需要考虑身份验证的问题，避免出现代码冗余等问题。本实例将介绍如何通过过滤器来进行用户身份验证。运行本实例，填写用户名和密码进行正常登录，则可以登录成功，当用户没有登录时，直接在 URL 地址栏输入 loginsuccee.jsp 页面，则会弹出提示信息，如图 7.6 所示。



图 7.6　未进行登录时显示提示信息

关键技术

实现本实例，主要是判断 session 域中是否存在用户名。首先，用户在登录时，系统会将用户名保存在 session 域中。在 Filter 过滤器的 doFilter()方法中，会判断 session 域中是否包含当前这个用户名，如果存在则允许登录；否则会弹出提示信息，并将当前的请求地址转向用户登录页。

设计过程

（1）创建过滤器的实现类 FilterLogin.java，先初始化 init(FilterConfig filterConfig)方法，再在执行体中判断 session 是否有 user 对象，如果没有则输出提示语句，否则继续执行，主要代码如下：

```
public class FilterLogin extends HttpServlet implements Filter {
    private FilterConfig filterConfig;
    public void init(FilterConfig filterConfig) throws ServletException {
        this.filterConfig = filterConfig;
    }
    public void doFilter(ServletRequest request, ServletResponse response,FilterChain filterChain)
            throws ServletException,IOException {
        HttpSession session=((HttpServletRequest)request).getSession();
        response.setCharacterEncoding("gb2312");    //响应客户端类型
```

```
            if(session.getAttribute("user")==null){          //判断 session 中是否有 user 这个对象
                PrintWriter out=response.getWriter();        //创建一个输出流
                //如果为空则通过 JavaScript 脚本输出提示并跳转到 index.jsp 页面
                out.print("<script language=javascript>alert('您还没有登录！！！');window.location.href='../index.jsp';</script>");
            }else{
                filterChain.doFilter(request, response);     //否则继续执行
            }
        }
        public void destroy() {
        }
}
```

（2）创建 loginresult.jsp 页面，用于在 user 对象的 session 中执行跳转到下一页面，代码如下：

```
<%
 request.setCharacterEncoding("gb2312");
 String name=request.getParameter("name");
 String password=request.getParameter("password");
        User user=new User();
        user.setUsername(name);
        user.setPassword(password);
        session.setAttribute("user",user);
 response.sendRedirect("filter/loginsuccee.jsp");
%>
```

（3）创建 JavaBean 类 User，存放登录名称与密码，并设置 getter 与 setter 方法。

（4）创建用户登录页面，关键代码如下：

```
<body><div align="center">使用过滤器身份验证
<form name="form" method="post" action="loginresult.jsp" onSubmit="return checkEmpty()">
<table width="220"   border="0" align="center">
   <tr>
      <td >用户名：</td>
      <td ><input name="name" type="text"></td>
   </tr>
   <tr>
      <td >密  码：</td>
      <td><input name="password" type="password"></td>
   </tr>
</table><br>
   <input type="submit" name="Submit" value="登录">
   <input type="submit" value="退出"/>
</form>
```

（5）在 web.xml 文件中配置如下：

```
<filter>
      <filter-name>filterUser</filter-name>
      <filter-class>com.mr.filter.FilterLogin</filter-class>
</filter>
<filter-mapping>
      <filter-name>filterUser</filter-name>
      <url-pattern>/filter/*</url-pattern>
</filter-mapping>
```

秘笈心法

在设置需要进行身份验证的页面时，应该把登录页面排除在外；否则，如果登录页面无法通过身份验证，用户将无法进行登录，所有进行身份验证的页面都将无法访问。

实例 186	字符编码过滤器 光盘位置：光盘\MR\07\186	初级 实用指数：★★

实例说明

初学者在编写 Web 程序时，经常出现页面编码混乱的问题，也就是中文字符显示成乱码。这类问题经常困

扰初学者，本实例使用 Servlet 过滤器解决编码导致的页面乱码问题。本实例使用 GBK 编码格式，如果不进行编码转换，将在页面中输出乱码；而通过过滤器自动将所有数据都转换为 GBK 编码就不会出现乱码，如图 7.7 所示。

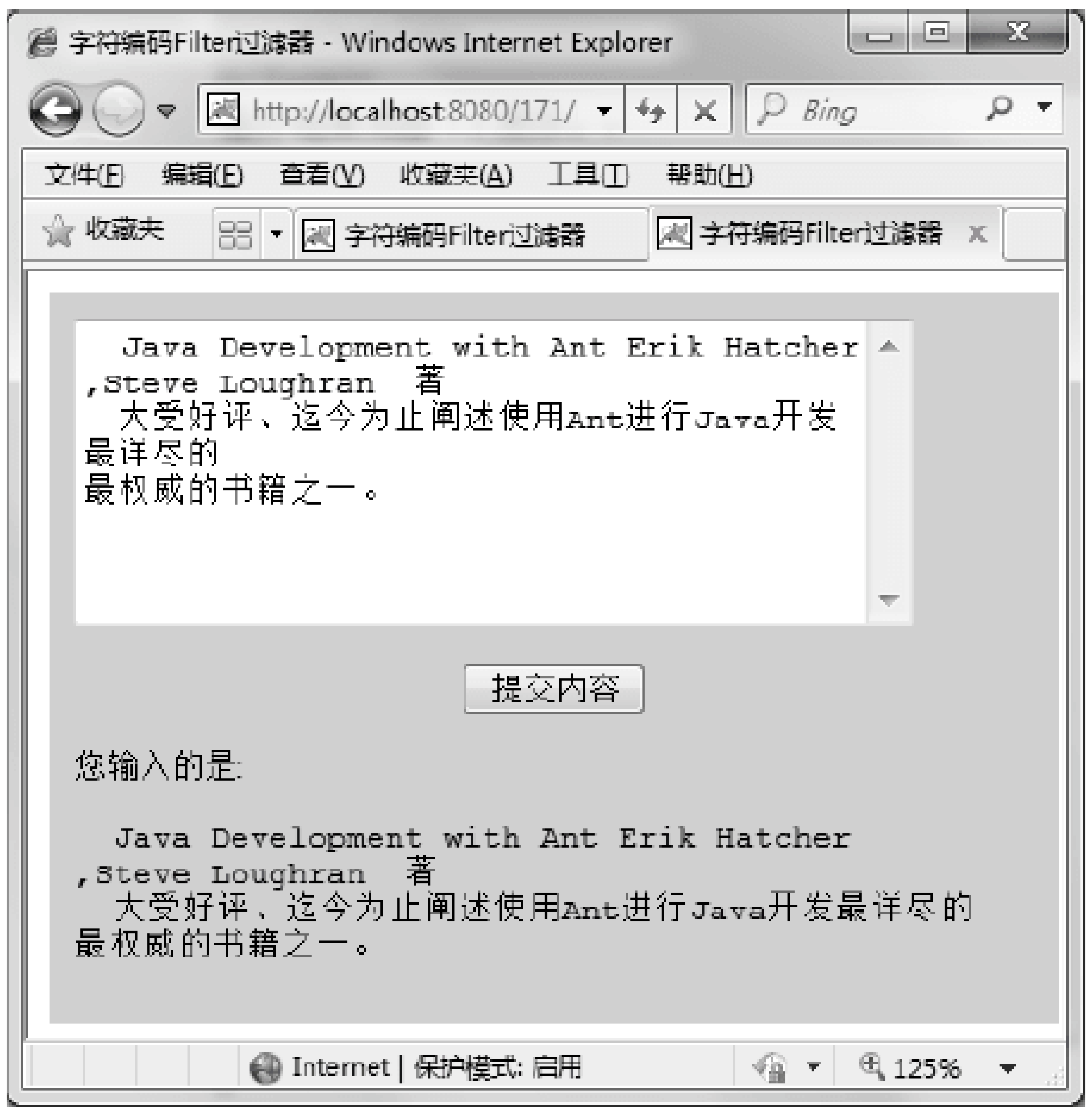


图 7.7　字符编码过滤器

关键技术

本实例主要在 web.xml 中配置默认的字符编码，并在 Filter 过滤器的初始化 init()方法中加载这个参数，然后在执行 doFilter()方法中获取这个编码的配置参数，将它设置为 request 和 response 的默认编码。

设计过程

（1）创建过滤器的实现类 Encoding.java，用于处理页面的字符编码格式。在初始化方法 init()中，加载 web.xml 配置的编码方式，并启用 enabled 编码，然后在 doFilter()方法中设置请求以及相应的编码格式，关键代码如下：

```
public class EncodingFilter implements Filter {
 private String Encoding;                              //配置 web.xml 编码
 private boolean enabled;                              //是否启用 Filter
 public void init(FilterConfig config) throws ServletException {
        Encoding = config.getInitParameter("Encoding");        //编码方式
        enabled = "true".equalsIgnoreCase(Encoding.trim())|| "1".equalsIgnoreCase(Encoding.trim());
 }
 public void doFilter(ServletRequest request, ServletResponse response,
                FilterChain chain) throws IOException, ServletException {
        if (enabled || Encoding != null) {                     //如果启用了此 Filter
                request.setCharacterEncoding(Encoding);        //request 的编码
                response.setCharacterEncoding(Encoding);       //response 的编码
        }
        chain.doFilter(request, response);                     //继续执行下一个 Filter
 }
 public void destroy() {
        Encoding = null;
}}
```

（2）在 web.xml 中配置初始化编码格式，主要代码如下：

```
<filter>
        <filter-name>encodingFilter</filter-name>
        <filter-class>com.mr.encoding.EncodingFilter</filter-class>
        <init-param>
                <param-name>Encoding</param-name>
                <param-value>UTF-8</param-value>
        </init-param>
        <init-param>
                <param-name>enabled</param-name>
```

```
                <param-value>true</param-value>
        </init-param>
</filter>
<filter-mapping>
        <filter-name>encodingFilter</filter-name>
        <url-pattern>/*</url-pattern>
</filter-mapping>
```

（3）创建 Encoding.jsp 页面，应用 EL 表达式输出编码后的内容，关键代码如下：

```
<form action="${ param.request.requestURL} " method="post">
 <table align="left" bgcolor="lightblue" cellpadding="3"
                        cellspacing="7">
        <tr><td>
           <textarea name="text" rows="8" cols="40">${ param.text }</textarea>
                </td>
        <tr><td align="center" colspan="4">
                        <input type="submit" value="提交内容" />
                </td></Tr>
    <tr><Td>您输入的是:<br/>
                <div><font size="6" >${ param.text }</font></div></Td></tr>
 </table>
</form>
```

秘笈心法

如果表单请求方式为 GET，还需要在 Tomcat 的文件夹下的 conf/server.xml 配置文件中修改 URIEncoding 参数，因为默认的编码为 ISO-8859-1，否则依然会出现乱码。

实例 187	使用过滤器监控网站流量 光盘位置：光盘\MR\07\187	初级 实用指数：★★

实例说明

Servlet 过滤器可以对用户提交的数据或服务器返回的数据进行更改。任何到达服务器的请求都会首先经过过滤器的处理。本实例应用过滤器的这个特点编写一个专门用于流量统计的过滤器，运行结果如图 7.8 所示。

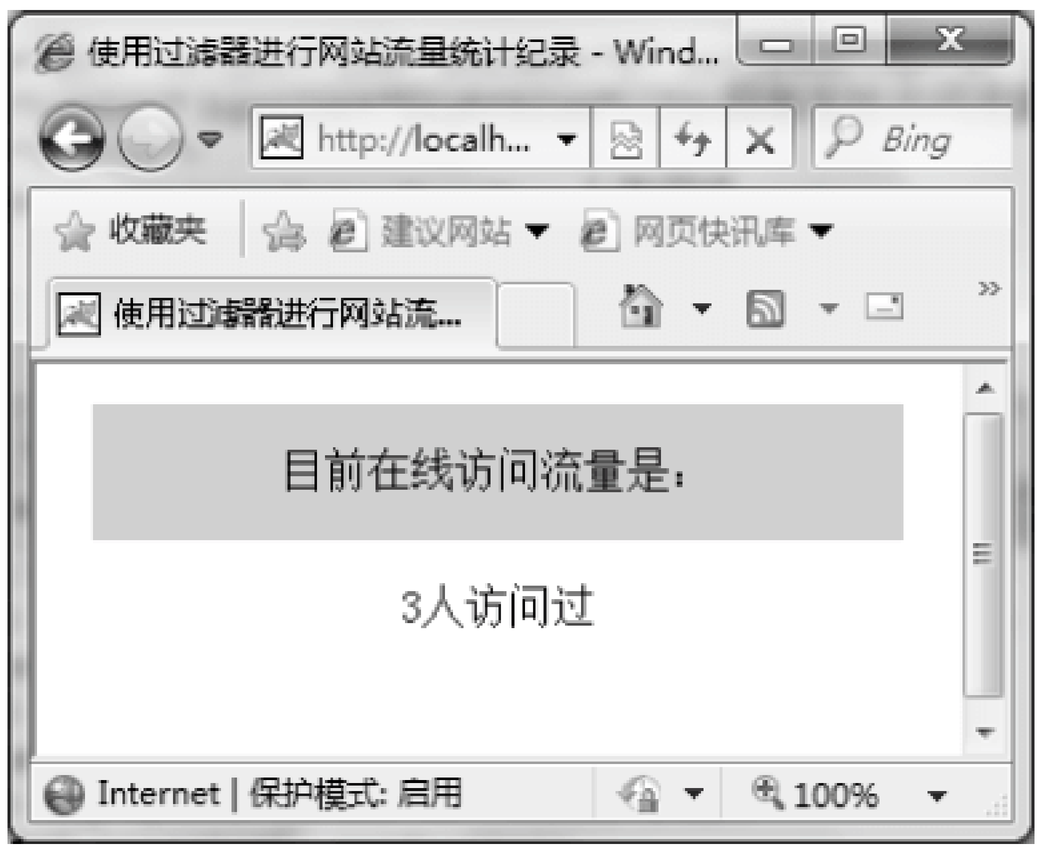


图 7.8　运行结果

关键技术

首先，需要一个整型变量来记录当前的流量，它需要在这个服务器中一直存在，可以用一个整型的静态私有变量来存放这个值，它的初值为 0，设置代码如下：

```
private static int flux = 0;
```

其次，在每次有用户访问页面时，必须对统计变量进行相应的修改。但是对于 Web 系统中多用户访问的系统，修改变量时必须考虑到同步的问题，即同一时间只能有一个用户修改这个整型变量，否则将会产生共享冲突，引起统计数据的不准确（得到的值比实际值小）。

在 Java 中可以通过关键字 synchronized 来解决这个问题。被 synchronized 关键字修饰的方法在执行过程中不会中断，也就是说线程一旦进入 synchronized 修饰的方法，其他线程就不会被阻塞，直到当前线程执行完这个方法为止。在本实例中就可以使用关键字 synchronized 来解决上述问题。

最后，把统计变量放入 request 中，以便任何页面都可以随时访问到网站当前的流量值，执行代码如下：

```
request.setAttribute("flux",String.valueOf(flux));
```

设计过程

（1）创建过滤器的实现类 FilterNum.java，通过 Servlet 中的过滤器技术统计网站的访问量，关键代码如下：

```
public class FilterNum extends HttpServlet implements Filter {
      private static int num = 0;    //定义全局变量
      public void init(FilterConfig filterConfig) throws ServletException {
      }
public synchronized void doFilter(ServletRequest request, ServletResponse response,FilterChain filterChain)
         throwsServletException, IOException {
          this.num++;                   //自增长
          request.setAttribute("num",String.valueOf(num));
          filterChain.doFilter(request, response);
      }
      public void destroy() {
      }
}
```

（2）创建 index.jsp 页面，主要代码如下：

```
<table width="300" height="100" border="0" cellpadding="0" cellspacing="0" >
         <tr align="center" bgcolor="lightblue"><Td>目前在线访问流量是：</Td></tr>
      <tr align="center">
      <td><%=request.getAttribute("num")%>人访问过</td>
      </tr>
</table>
```

（3）在 web.xml 中配置 FilterNum 过滤器，关键代码如下：

```
<filter>
      <filter-name>filterNum</filter-name>
      <filter-class>com.mr.filter.FilterNum</filter-class>
</filter>
   <filter-mapping>
      <filter-name>filterNum</filter-name>
      <url-pattern>/*</url-pattern>
</filter-mapping>
```

秘笈心法

为了避免多个用户并发访问而导致数据的不准确，可以使用 synchronized 关键字修饰方法。

实例 188	防止页面缓存的过滤器 光盘位置：光盘\MR\07\188	初级 实用指数：★★

实例说明

IE 缓存虽然能够提高已存储网站的访问速度，但是过度的 IE 缓存会影响浏览器的响应速度。同时还可能为网站的运行带来一些不必要的麻烦。例如，可能会因为浏览器缓存的应用，而导致 Web 服务器不能准确地计算一个页面或广告被阅览的次数；在论坛或者网上商城系统中由于浏览器缓存的使用，导致更新的图片信息不能得到及时的显示。这些都是浏览器缓存带来的负面影响。本实例主要介绍浏览网页时不自动缓存，这样下次访问能及时更新图片或者相关信息，运行结果如图 7.9 所示。

关键技术

本实例主要应用过滤器（Filter）防止页面缓存。关键是应用 javax.servlet.Filter 接口中提供的 doFilter()方法。

在 doFilter()方法中设置 HTML 中 meta 标签的 http-equiv 属性，实现禁止浏览器缓存的功能。

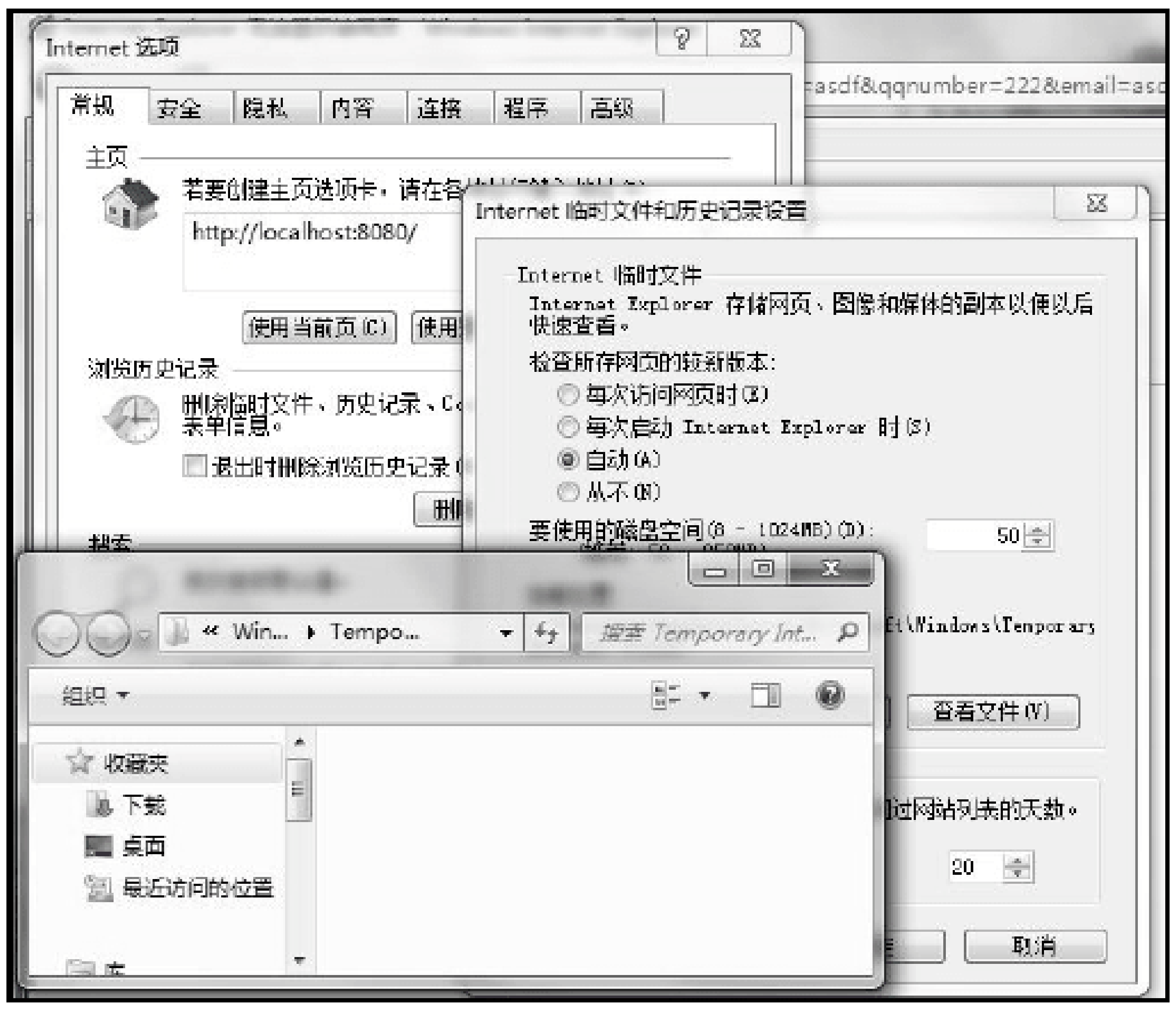


图 7.9　缓存文件夹为空

（1）设置 http-equiv 属性的参数 expires，控制网页的过期时间。

（2）设置 http-equiv 属性的参数 Pragma，禁止浏览器从本地计算机的缓存中访问页面内容。

（3）设置 HTTP 消息头中的 Cache-control 参数，控制页面的缓存。Cache-control 的常见值有 private、no-cache、max-age 和 must-revalidate 等，默认值为 private。

Cache-control 的作用根据浏览方法的不同可以分为以下几种情况：

（1）以打开新窗口的方式进行浏览

如果指定 Cache-control 的值为 private、no-cache 或者 must-revalidate，那么打开新窗口访问时就会重新访问服务器；如果指定的值为 max-age，那么在此值规定的时间中就不会重新访问服务器，如 Cache-control: max-age=10 表示当访问此网页后的 10 秒内不会再次访问服务器。

（2）在地址栏中按 Enter 键进行浏览

如果值为 private 或 must-revalidate，则只有第一次访问时会访问服务器，以后就不再访问；如果值为 no-cache，那么每次都会访问；如果值为 max-age，则在过期之前不会重复访问。

（3）按后退键进行浏览

如果值为 private、must-revalidate 或 max-age，则不会重复访问；如果值为 no-cache，则每次都重复访问。

（4）按刷新键

无论为何值，都会重复访问。如果指定 Cache-control 值为 no-cache 时，访问此页面不会在 Internet 临时文件夹中留下页面备份。

设计过程

（1）创建过滤器的实现类 NoCacheFilter.java，设置 HTTP 信息头禁止浏览器从缓存中读取页面。在 doFilter()方法中通过设置响应头信息防止页面缓存，关键代码如下：

```
public class NoCacheFilter implements Filter {
public void doFilter(ServletRequest request, ServletResponse response, FilterChain filterchain) throws IOException, ServletException {
        //HTTP 消息头控制网页的缓存
    ((HttpServletResponse) response).setHeader("Cache-Control","no-cache");
        //禁止浏览器从本机的缓存中读取页面
((HttpServletResponse)response).setHeader("Pragma","no-cache");
((HttpServletResponse) response).setHeader("Expires", "-1");      //缓存中的有效期
            filterchain.doFilter(request, response);
    }
    public void destroy() {
    }
}
```

（2）在 web.xml 中配置 NoCacheFilter.java 过滤器，关键代码如下：

```
<filter>
        <filter-name>BrowserNoCacheFilter</filter-name>
        <filter-class>com.mr.nocache.NoCacheFilter</filter-class>
        <init-param>
            <param-name>Cache-control</param-name>
            <param-value>no-cache</param-value>
        </init-param>
        <init-param>
            <param-name>Pragma</param-name>
            <param-value>no-cache</param-value>
        </init-param>
        <init-param>
            <param-name>Expires</param-name>
            <param-value>-1</param-value>
        </init-param>
</filter>
<filter-mapping>
        <filter-name>NoCacheFilter</filter-name>
        <url-pattern>/*</url-pattern>
        <dispatcher>REQUEST</dispatcher>
        <dispatcher>FORWARD</dispatcher>
</filter-mapping>
<session-config>
        <session-timeout>30</session-timeout>
</session-config>
```

提示：Session-timeout 为缓存有效期 30 秒。

秘笈心法

缓存文件夹的打开方式是，选择 IE 浏览器，单击鼠标右键，并在弹出的快捷菜单中选择“属性”命令，弹出 Internet 属性对话框，选择“常规”选项卡，在 Internet 监时文件中单击“设置”按钮，在弹出的设置对话框中单击“查看文件”按钮，将看到浏览器缓存中存储的文件，而此时该文件夹下没有任何文件。

实例 189　通过过滤器控制页面输出内容

光盘位置：光盘\MR\07\189

初级　实用指数：★★

实例说明

在开发程序的过程中，很多程序员都实现过在打开的页面中弹出一个对话框的功能，通过该对话框来输出一些广告或者公告信息。如果想将这个功能应用到多个页面中，就需要在多个页面中添加调用弹出对话框功能的代码来完成，这样做虽然能够实现此功能，但是却增加了代码的冗余。

本实例将介绍一种不必在多个页面中编写代码而实现弹出对话框的方法，即通过过滤器来控制页面输出的内容，进而实现在每个响应的页面中都弹出一个对话框的功能。运行本实例，单击页面中的任何超链接，都会弹出一个新的对话框，如图 7.10 所示。

关键技术

本实例的关键在于在完成过滤任务时，将请求的对象返回到自定义的应答对象中，通过自定义应答对象对请求的数据进行编译，编译完成后通过自定义的方法返回响应数据，并通过 replace()方法向响应的数据中添加调用弹出对话框的代码。完成通过过滤器控制页面输出的操作。

replace()方法的语法结构如下：

```
public String replace(char oldChar, char newChar)
```

将该方法中的 newChar 替换指定字符串中出现的所有 oldChar，返回一个新的字符串。

参数说明

❶ oldChar：要替换的子字符串或者字符。

❷ newChar：新的字符串或字符，用于替换原有字符串的内容。



图 7.10　转换后的效果

设计过程

（1）创建 OutputStream.java 文件并且继承 ServletOutputStream 类，替换父类的输出流，关键代码如下：

```
public class OutputStream extends ServletOutputStream {
    ByteArrayOutputStream stream;                         //创建字节数组输出流
    public OutputStream(ByteArrayOutputStream stream) {
        this.stream = stream;                             //构造方法初始化输出流
    }
    public void write(int b) throws IOException {
        stream.write(b);                                  //使用此类的输出流替换父类的输出方法
    }
}
```

（2）创建 ResponseWrapper.java 文件并且继承 HttpServletResponseWrapper，使响应对象进行重新编译并返回响应的数据，关键代码如下：

```
public class ResponseWrapper extends HttpServletResponseWrapper {
    private OutputStream stream;                          //声明一个输出流
    private ByteArrayOutputStream byteStream;             //声明字节数组输出流
    private PrintWriter pw;                               //声明打印输出流
    public ResponseWrapper(HttpServletResponse response) {
        super(response);
        byteStream = new ByteArrayOutputStream();         //数据流初始化
        stream = new OutputStream(byteStream);
        pw = new PrintWriter(byteStream);
    }
    public ServletOutputStream getOutputStream() throws IOException {
        return stream;                                    //返回字节输出流并重写父类方法
    }
    public PrintWriter getWriter() throws IOException {
        return pw;                                        //返回打印输出流重写父类方法
    }
    public String getContent() throws UnsupportedEncodingException {
        return byteStream.toString();                     //返回响应数据
    }
}
```

（3）创建过滤器的实现类 OutFilter.java，在 doFilter()方法中完成对过滤器的操作，并用 getContent()获取

到响应的数据，用 replace()方法将弹出对话框的代码层加到 response 响应的数据中去，关键代码如下：

```
public class OutFilter implements Filter {
    private boolean variable=true;                //如果 variable 为真，每次都生成 HTML 首页

    private FilterConfig filterConfig = null;

    public void init(FilterConfig filterConfig) {
        this.filterConfig = filterConfig;
    }
    public void doFilter(ServletRequest request, ServletResponse response,
            FilterChain chain)
            throws IOException, ServletException {
        HttpServletResponse httpResp = (HttpServletResponse) response;

        ResponseWrapper responseWrapper = new ResponseWrapper(httpResp);
        chain.doFilter(request, responseWrapper);    //过滤器的操作
        PrintWriter out = response.getWriter();      //创建输出流
        responseWrapper.getWriter().flush();         //获取输出流并强制刷新
        String str=responseWrapper.getContent();
        String
stres="</head><script>window.open('message.htm','','width='+300+',height='+180+',top='+'+window.screen.width-300+'+',left='+'+window.screen.heigh
t+180+');</script>";
        out.println(str.replace("</head>",stres));
    }
    public void destroy() {
    }
    public void log(String msg) {
        filterConfig.getServletContext().log(msg);
    }
}
```

（4）在 web.xml 中配置过滤器，关键代码如下：

```
<filter>
  <filter-name>CharacterEncodingFilter</filter-name>
  <filter-class>com.mr.filter.CharacterEncodingFilter</filter-class>
  <init-param>
    <param-name>encoding</param-name>
    <param-value>GBK</param-value>
  </init-param>
 </filter>
<filter>
  <filter-name>outFilter</filter-name>
  <filter-class>com.mr.filter.OutFilter</filter-class>
 </filter>
 <filter-mapping>
  <filter-name>CharacterEncodingFilter</filter-name>
  <url-pattern>/*</url-pattern>
  <dispatcher>REQUEST</dispatcher>
  <dispatcher>FORWARD</dispatcher>
 </filter-mapping>
 <filter-mapping>
  <filter-name>outFilter</filter-name>
  <url-pattern>/index.jsp</url-pattern>
  <dispatcher>REQUEST</dispatcher>
  <dispatcher>FORWARD</dispatcher>
</filter-mapping>
    <filter-mapping>
        <filter-name>outFilter</filter-name>
        <url-pattern>/indexsure.jsp</url-pattern>
        <dispatcher>REQUEST</dispatcher>
        <dispatcher>FORWARD</dispatcher>
    </filter-mapping>
```

秘笈心法

在开发 Web 工程时，网页中弹出一个对话框是很常见的，如果想在多个页面中添加此功能，则每个页面都添加对话框功能的代码会带来大量代码的冗余，本实例正是解决此办法之一。

实例 190 使用过滤器自动生成静态页面

光盘位置：光盘\MR\07\190

初级

实用指数：★★

实例说明

在开发程序的过程中，创建的几乎都是动态页面，因为在每个页面中都要使用或者传递一些数据，所以在浏览网页时速度就不是很理想，总是因为数据的处理而影响网页打开的速度。然而，如果都是静态网页，网页的打开速度就不会受到影响，同样还可以更改网页文件的后缀名，从而隐藏程序开发使用的语言。本实例将介绍一种生成静态网页的方法，即通过过滤器将动态页面生成静态页，运行结果如图 7.11 所示。



图 7.11 自动生成静态页面

关键技术

通过过滤器生成静态页面主要应用过滤器接受请求，然后通过自定义的响应对象对请求的数据进行处理，生成一个静态页面，最后将经过处理的响应数据返回到客户端。

首先实现 Filter 接口，然后定义 doFilter()方法，在该方法中获取到请求的数据，应用自定义的响应对象 ResponseWrapper 对请求数据进行过滤，过滤业务完成后生成静态页面。

设计过程

（1）创建 OutputStream 类，在构造方法中初始化输出流，定义 writer()方法并替换父类的输出方法，关键代码如下：

```
public class OutputStream extends ServletOutputStream {
    ByteArrayOutputStream ostream;                              //创建字节数组输出流
    public OutputStream(ByteArrayOutputStream ostream) {
        this.ostream = ostream;                                 //在构造方法中初始化输出流
    }
    public void write(int b) throws IOException {
        ostream.write(b);                                       //使用此类输出流替换父类的输出方法
    }
}
```

（2）创建 ResponseWrapper 类文件，该类继承 HttpServletResponseWrapper 类，使用它对请求的数据进行处理，关键代码如下：

```
public class ResponseWrapper extends HttpServletResponseWrapper {
    private OutputStream ostream;                               //自定义输出流
```

```
    private ByteArrayOutputStream byteStream;                    //字节数组输出流
    private PrintWriter pw;                                      //打印输出流
    public ResponseWrapper(HttpServletResponse response) {
        super(response);
        byteStream = new ByteArrayOutputStream();                //数据流初始化
        ostream = new OutputStream(byteStream);
        pw = new PrintWriter(byteStream);
    }
    public ServletOutputStream getOutputStream() throws IOException {
        return ostream;                                          //返回字节输出流并重写父类方法
    }
    public PrintWriter getWriter() throws IOException {
        return pw;                                               //返回打印输出流重写分类方法
    }
    public String getContent() throws UnsupportedEncodingException {
        return byteStream.toString();                            //自定义方法返回响应的数据
    }
}
```

（3）创建过滤器的实现类 StaticHtmlFilter，将定义的对象为响应对象来完成过滤操作，在定义的响应对象中获取请求的数据，并对其进行修改生成静态页面，最后将修改后的响应返回到客户端，关键代码如下：

```
public class StaticHtmlFilter implements Filter {
    private boolean variable = true;                             //定义一个变量 variable 时，每次都生成 HTML 首页
    String strs = "";
    private FilterConfig filterConfig = null;
    public void init(FilterConfig filterConfig) {
        this.filterConfig = filterConfig;
    }
    public void doFilter(ServletRequest request, ServletResponse response,FilterChain chain)
            throws IOException, ServletException {
        HttpServletResponse httpRes = (HttpServletResponse) response;
        String path = ((HttpServletRequest) request).getServletPath();
        strs = path.substring(1);                                //获取文件名
        ResponseWrapper responseWrapper = new ResponseWrapper(httpRes);   //创建定义的应答对象
        chain.doFilter(request, responseWrapper);                //完成过滤的操作
        responseWrapper.getWriter().flush();
        int len = strs.length();                                 //获取文件名长度
        String str = strs.substring(0, len - 4) + ".html";       //定义静态页文件的名称
        String jspPath = request.getRealPath(str);               //获取文件的真实存储路径
        File jspFile = new File(jspPath);                        //创建页面文件对象
        String static_path = jspFile.getParent() + "\\";         //定义静态页文件存储的位置
        File htmlFile = new File(static_path, str);              //创建静态页文件对象
        Date htmlDate = null;
        Date now = new Date();
        if (htmlFile.exists()) {
            htmlDate = new Date(htmlFile.lastModified());
        } else {
            htmlFile.createNewFile();
        }
        //如果已生成的 HTML 文件不是当前的，就重新生成一个
        if (variable || htmlDate == null || htmlDate.getDate() != now.getDate()) {
            FileOutputStream fileStream = new FileOutputStream(htmlFile);     //创建 HTML 文件的输出流
            DataOutputStream fout = new DataOutputStream(fileStream);        //创建数据输出流
            DateFormat dateFormat = DateFormat.getDateTimeInstance(FULL, FULL);//创建日期格式器
            fout.writeChars(" ");                                //输出一个空字符
            fout.writeUTF("生成时间：" + dateFormat.format(new Date()));     //使用 UTF 格式输出注释信息，标注 HTML 文档生成事件
            fout.writeUTF(responseWrapper.getContent());         //使用 UTF 格式输出 HTML 内容，保存到 HTML 文件中
            fout.close();
        }
        System.out.println("/"+str+"");               //如果当天已经生成过 HTML 文件，直接把请求转发给 HTML 文件
        request.getRequestDispatcher("/"+str+"").forward(request,response);
    }
    public void destroy() {
    }
    public void log(String msg) {
```

```
            filterConfig.getServletContext().log(msg);
        }
}
```

秘笈心法

本实例使用的 ResponseWrapper 对象是自定义 Java 类文件，用于对请求的数据流进行处理，其中还应用了自定义的 OutputStream.java 类对父类的输出方法进行替换。

实例 191　文件上传过滤器

光盘位置：光盘\MR\07\191

初级

实用指数：★★

实例说明

本实例主要把上传的文件保存到系统文件夹中，然后通过自定义的 request 传入 Filter，再使用 getAttribute() 方法直接获取上传的文件，运行效果如图 7.12 所示。



图 7.12　上传结果

关键技术

实现本实例时，在上传文件时需要把<form>标签的 enctype 属性设置为 multipart/form-data，然后创建一个继承自 HttpServletRequestWrapper 类的自定义请求，在该自定义请求中，转换由过滤器的 doFilter()方法传递过来的原始请求，然后在这个自定义的请求中应用 Apache 的文件上传组件对原始请求进行解析，判断是否为上传文件的请求，如果是文件则保存到服务器的指定目录，并将解析出的上传文件对象保存到 Map 中。

设计过程

（1）创建实现类 UploadFilter.java，自定义一个 request 处理文件上传，主要代码如下：

```
public class UploadFilter implements Filter {
public void destroy() {
}
public void doFilter(ServletRequest request, ServletResponse response,
            FilterChain chain) throws IOException, ServletException {
                                    //自定义上传的类型
    UploadRw uploadRequest = new UploadRw(
            (HttpServletRequest) request);
```

```
        chain.doFilter(uploadRequest, response);                          //继续执行下一个 Filter
            }
 Public void init(FilterConfig filterConfig) throws ServletException {
 }
}
```

（2）创建 UploadRw.java 类文件来处理文件读取以及通过 apache 工具来解析文件，然后通过 map 取值输出到文件中去，主要代码如下：

```
public class UploadRw extends HttpServletRequestWrapper {
 private static final String MULTIPART_HEADER = "Content-type";          //文件类型
 private boolean multipart;                                                //确认上传文件
                                                                            //保存提交的数据
 private Map<String, Object> param = new HashMap<String, Object>();
 @SuppressWarnings("all")
 public UploadRw(HttpServletRequest request) {
        super(request);
                                                                            //判断是否为上传文件
        multipart = request.getHeader(MULTIPART_HEADER)!= null
                    && request.getHeader(MULTIPART_HEADER).startsWith(
                            "multipart/form-data");
        if (multipart){
            try {
                                                                            //使用 apache 自带的工具解析
                    DiskFileUpload upload = new DiskFileUpload();
                    upload.setHeaderEncoding("utf8");
                                                                            //获得所有的文本域与文件域
                    List<FileItem> fileItems = upload.parseRequest(request);
                    for (Iterator<FileItem> it = fileItems.iterator(); it.hasNext();) {
                                                                            //使用遍历
                        FileItem item = it.next();
                        if (item.isFormField()) {
                                                                            //如果是文本域直接放到 Map 中
                            param.put(item.getFieldName(), item.getString("utf8"));
                        } else {
                                                                            //否则为文件先获取文件名称
                            String filename = item.getName().replace("\\", "/");
                                                                            //替换特殊字符或字符串
                            filename = filename.substring(filename.lastIndexOf("/") + 1);
                                                                            //保存到系统临时文件夹中
                            File file = new File(System.getProperty("java.io.tmpdir"), filename);
                                                                            //保存文件内容
                            OutputStream ous = new FileOutputStream(file);
                            ous.write(item.get());
                            ous.close();
                                                                            //放入 map 中
                            param.put(item.getFieldName(), file);
                        }
                    }
            } catch (Exception e) {
                    e.printStackTrace();
            }
        }
 }
 public Object getAttribute(String name) {
                                                                            //如果是上传文件就到 map 中取值
        if (multipart && param.containsKey(name)) {
            return param.get(name);
        }
        return super.getAttribute(name);
 }
```

（3）创建 upload.jsp 页面，主要代码如下：

```
<div align="center">
            信息头是 = ${ header['Content-type'] }
</div>
        <br />
<form action="" method="post" enctype="multipart/form-data">
```

```
        <table align="center" cellpadding="0" cellspacing="0" background="images/upload.jpg" height="400" width="500">
    <tr height="50" align="center">
                <td width="250">
                        <input type="file" name="file1" border="2" value="浏览">
                </td>
<%
File file1 = (File) request.getAttribute("file1");
        if (file1 != null)
                out.println("<br/>文件: " + file1 + ", <br/>大小字节 ：" + file1.length());
%>
</tr>
<Tr align="center" height="50">
                <td width="250">
                        <input type="submit" value=" 上传文件 ">
                </td>
</Tr>
<Tr height="200">
                <td> </td>
</Tr>
        </table>
</form>
```

注意：本实例中自定义了 request 方法，需要添加 apache 的 common-fileuploadin.jar 的架包（在本实例 Web 工程包的 webRoot\web-INF\lib 目录下可以找到）。

秘笈心法

本实例为简单起见，自定义的 request 并没有覆盖 getParameterNames 等方法，也没有处理多选框提交相同数据的情况。

实例 192　权限验证过滤器　初级

光盘位置：光盘\MR\07\192　实用指数：★★

实例说明

本实例实现对 session 的校验，如果没有相对应的处理就抛出一个 LoginException 异常。本实例添加一个 URI 与权限 role 角色检查，这个配置文件存放在 properties 配置文件中，运行结果如图 7.13 所示。



图 7.13　左为允许访问（右为无权限访问）

关键技术

本实例主要用到 properties 配置文件来保存所有的权限，然后应用 Properties 类来操作 properties 文件。Properties 类表示了一个持久的属性集。配置文件 properties 可保存在流中或从流中加载。属性列表中每个键及

其对应值都是一个字符串，在设置访问路径时用到 RequestURI 方法获取路径，再获取相应的参数，本实例为 action，这两个组合在一起形成一个新的 URI。

设计过程

（1）创建过滤器的实现类 PriorityFilter.java，在该类中创建一个 Properties 对象，使它可以保存在流中或从流中加载，作用是保存所有的权限，并在初始化方法中获取这个权限文件的位置与配置，在 doFilter()中设置访问的路径与后缀的参数，并组成一个新的 URI，主要代码如下：

```
public class PriorityFilter implements Filter {
 private Properties pts = new Properties();
 public void init(FilterConfig config) throws ServletException {
        //从初始化参数中获取权限配置文件的位置
        String file = config.getInitParameter("file");
        String realPath = config.getServletContext().getRealPath(file);
        try {
                pts.load(new FileInputStream(realPath));
        } catch (Exception e) {
                config.getServletContext().log("读取权限文件错误", e);
        }
 }
 public void doFilter(ServletRequest req, ServletResponse res,
                FilterChain chain) throws IOException, ServletException {
        HttpServletRequest request = (HttpServletRequest) req;
        //获取访问的路径
        String requestURI = request.getRequestURI().replace(request.getContextPath() + "/", "");
        //获取  action 的参数
        String action = req.getParameter("action");
        action = action == null ? "" : action;
        //组合成新的  URI
        String uri = requestURI + "?action=" + action;
        //在 session 中获取用户权限
        String role = (String) request.getSession(true).getAttribute("role");
        role = role == null ? "guest" : role;
        boolean authentificated = false;
        //审核用户是否有权限登录访问
        for (Object obj : pts.keySet()) {
                String key = ((String) obj);
                //使用正则表达式验证，需要将 ？ 替换，并通过通配符 * 处理
                if (uri.matches(key.replace("?", "\\?").replace(".", "\\.").replace("*", ".*"))) {
                        //如果 role 角色匹配
                        if (role.equals(pts.get(key))) {
                                authentificated = true;
                                break;
                        }
                }
        }
        if (!authentificated) {
                throw new RuntimeException(new LoginException("您无权访问该页面。请以合适的身份登录后查看。"));
        }
        //下一个过滤器或者 Servlet
        chain.doFilter(req, res);
 }
 public void destroy() {
        pts = null;
 }
}
```

（2）创建 priority.properties 配置文件，如果只有 key-value 属性值，其中 key 键为访问的地址，value 为控制访问的权限名称，主要代码如下：

```
# Privilege Settings

admin.do?action\=*=administrators
login.do?action\=*=administrators
method.do?action\=add=system
method.do?action\=delete= system
```

```
method.do?action\=save = system
method.do?action\=view = guest
method.do?action\=list = guest
```

（3）配置 web.xml 文件，其中捕捉异常要用到上个异常过滤器的范例，这样拿过来直接调用即可，主要代码如下：

```
<display-name>filter</display-name>
<servlet>
        <servlet-name>dispatcherServlet</servlet-name>
        <jsp-file>/output.jsp</jsp-file>
</servlet>
<servlet-mapping>
        <servlet-name>dispatcherServlet</servlet-name>
        <url-pattern>*.do</url-pattern>
</servlet-mapping>
        <filter>
        <filter-name>exceptionFilter</filter-name>
        <filter-class>
                com.mr.filter.ExceptionFilter
        </filter-class>
</filter>
<filter>
        <filter-name>priorityFilter</filter-name>
        <filter-class>
                com.mr.filter.PriorityFilter
        </filter-class>
        <init-param>
                <param-name>file</param-name>
                <param-value>/WEB-INF/priority.properties</param-value>
        </init-param>
</filter>
        <filter-mapping>
        <filter-name>exceptionFilter</filter-name>
        <url-pattern>/*</url-pattern>
</filter-mapping>
<filter-mapping>
<filter-name>priorityFilter</filter-name>
        <url-pattern>*.do</url-pattern>
</filter-mapping>
```

秘笈心法

正则表达式是一个描述字符模式的对象，在 Web 开发中正则表达式应用相当广泛，使用正则表达式可以检查一个字符串中是否含有指定的子字符串、替换匹配的子字符串为指定的内容或者从某个字符串中取出符合条件的子字符串等。

7.2 监听器的应用

Servlet 规范的另一个高级特性就是监听器，用于监听 Java Web 程序并触发相应的事件。监听器也是 Servlet 2.3 规范中加入的，事件发生时会自动触发对应的 Listener。主要用于监听的对象有 session、request、context 等。

实例 193	监听在线用户 光盘位置：光盘\MR\07\193	初级 实用指数：★★

实例说明

监听器的作用是监听 Web 容器的有效事件，它由 Servlet 容器管理，应用 Listener 接口监听某个执行程序，并根据该程序的需求做出适当的响应。本实例将通过监听器查看用户的在线情况，实例运行效果如图 7.14 所示。

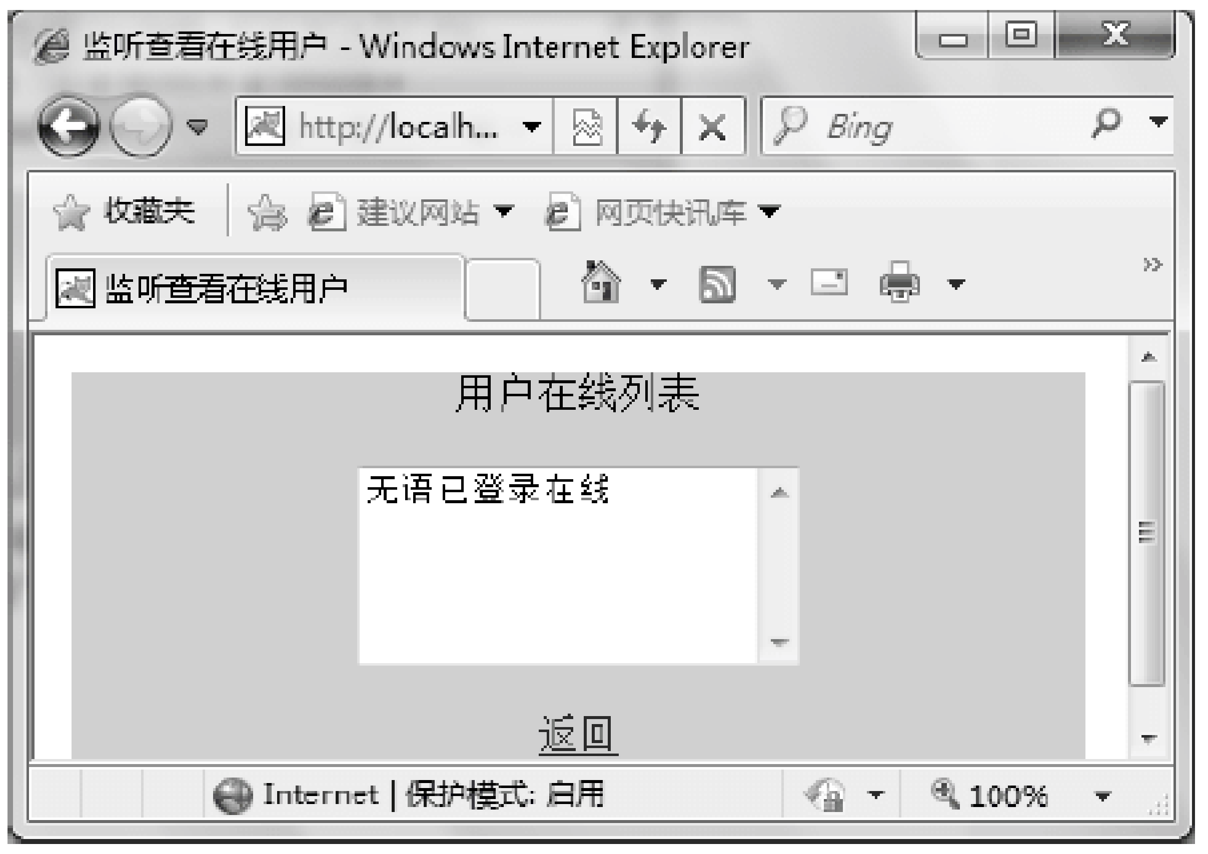


图 7.14　在线用户

关键技术

本实例中应用 HttpBindingListener 监听接口，它监听 HTTP 会话中对象的绑定信息。它是唯一不需要在 web.xml 中设定的监听。HttpBindingListener 接口提供以下两个方法。

- valueBound(HttpSessionBindingEvent arg0)：当有对象加入 session 的范围时会被自动调用。
- valueUnbound(HttpSessionBindingEvent arg0)：当有对象从 session 的范围内移除时会被自动调用。

设计过程

（1）创建 LoginList.java 类来存放用户和在线用户的具体操作，主要代码如下：

```
public class LoginList {
    private static LoginList user = new LoginList();
    private Vector vector = null;
    public LoginList() {
        this.vector = new Vector();
    }
     public static LoginList getInstance() {
        return user;
    }
    public boolean addLoginList(String user) {                //用户登录
        if (user != null) {
            this.vector.add(user);
            return true;
        } else {
            return false;
        }
    }
    public Vector getList() {                                 //获取用户列表
        return vector;
    }
    public void removeLoginList(String user) {                //删除用户
        if (user != null) {
            vector.removeElement(user);
        }
}
}
```

（2）创建 LoginNote.java 类，实现 HttpSessionBindingListener 类，关键代码如下：

```
public class LoginNote implements javax.servlet.http.HttpSessionBindingListener {
   private String user;
   private LoginList container = LoginList.getInstance();
   public LoginNote() {
      user = "";
   }
   public void setUser(String user) {
      this.user = user;
   }
   public String getUser() {
      return this.user;
   }
```

```
  public void valueBound(HttpSessionBindingEvent arg0) {
     System.out.println(this.user+"该用户已经上线" );
  }
  public void valueUnbound(HttpSessionBindingEvent arg0) {
     System.out.println(this.user+"该用户已经下线");
     if (user != "") {
     container.removeLoginList(user);
     }
  }
}
```

（3）创建 LoginList.jsp 在线用户页面，关键代码如下：

```
<%
LoginList list=LoginList.getInstance();
LoginNote ut=new LoginNote();
String name=request.getParameter("user");
ut.setUser(name);
session.setAttribute("list",ut);
list.addLoginList(ut.getUser());
session.setMaxInactiveInterval(10);
%>
<body>
<div align="center">
<table width="400" height="150" border="0" cellpadding="0" cellspacing="0" bgcolor="lightblue">
 <tr align="center"><td>用户在线列表</td></tr>
   <tr>
     <td align="center"><br>
  <textarea rows="5" cols="22">
<%
Vector vector=list.getList();
if(vector!=null&&vector.size()>0){
for(int i=0;i<vector.size();i++){
  out.println(vector.elementAt(i)+"已登录在线");
}
}
%>
</textarea><br><br>
  <a href="loginOut.jsp">返回</a>
```

当调用 valueBound 和 valueUnbound 时控制监听到用户的动作，如图 7.15 所示。

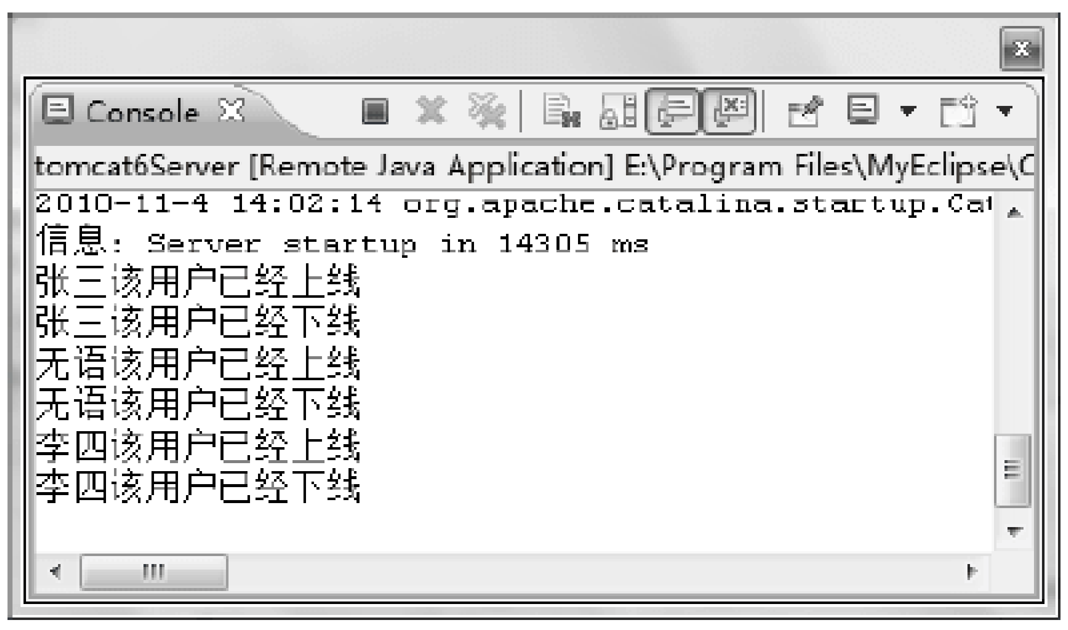


图 7.15　用户在线情况

秘笈心法

HttpBindingListener 接口，它监听 HTTP 会话中对象绑定的信息，也是唯一不需要在 web.xml 容器中设定的监听器。

实例 194　应用监听器使服务器端免登录

光盘位置：光盘\MR\07\194

初级

实用指数：★★

实例说明

对请求的监听是在 Servlet 2.4 规范中新增加的一个技术，当用户在监听程序中获得请求时，就可以对请求

进行统一的处理。本实例对指定的 IP 实现免登录的操作，如果是第一次登录，需要添加 IP 地址才能进入主页；如果不是第一次登录，说明该 IP 地址已经被添加到免登录 IP 数据库中，则可以直接进入到主页面，运行效果如图 7.16 所示。

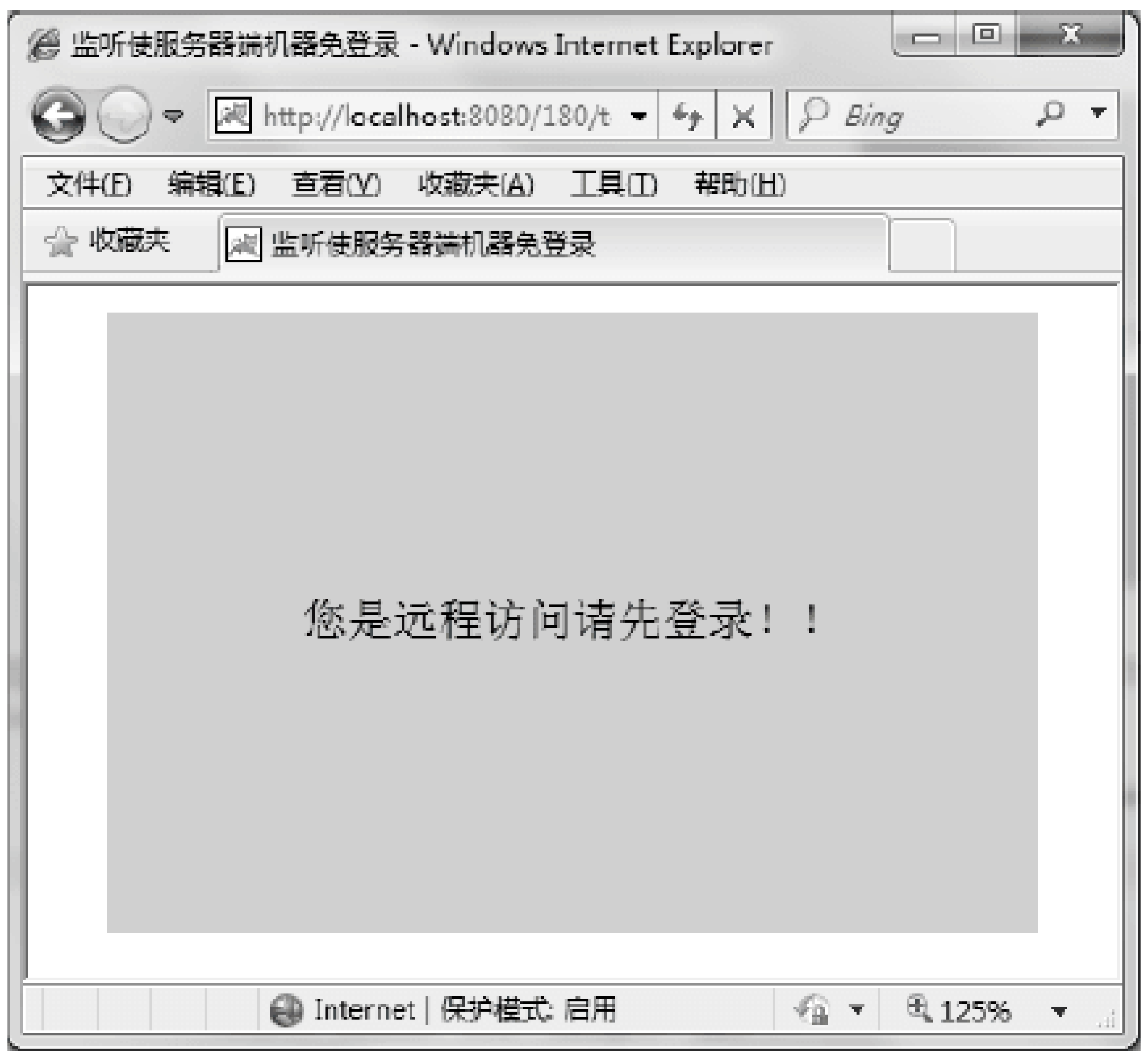


图 7.16　访问结果

关键技术

在本实例中实现客户端的请求和请求参数设置的监听需要实现以下两个接口。

（1）ServletRequestListener 接口

该接口提供了以下两个方法。

- requestInitialized(ServletRequestEvent sre)方法：通知正在收听的对象，ServletRequest 已经加载及初始化。
- requestDestroyed(ServletRequestEvent sre)方法：通知正在使用收听的对象，ServletRequest 已经被载出。

（2）ServletRequestAttributeListener 接口

该接口提供了以下 3 个方法。

- attributeAdded(ServletRequestAttributeEvent srae)方法：若有对象加入 Request 的范围时，通知正在收听的对象。
- attributeRemoved(ServletRequestAttributeEvent srae)方法：若在 Request 的范围内有对象取代另一个对象时，通知正在收听的对象。
- attributeReplaced(ServletRequestAttributeEvent srae)方法：若有对象从 Request 的范围移除时，通知正在收听的对象。

设计过程

（1）创建监听器类 LoginListener.java，使用 requestInitialized()方法获取请求的 IP 地址，根据 Login 对象的值，判断是否是服务器访问，此方法会向请求中增加 login 对象，再依据对象赋予不同的值，主要代码如下：

```
public class LoginListener extends HttpServlet implements
        ServletRequestListener, ServletRequestAttributeListener {
    public void requestInitialized(ServletRequestEvent sre) {
        System.out.println("已经初始化");
        ServletRequest sr = sre.getServletRequest();
        System.out.println("远程访问机器的 IP:" + sr.getRemoteAddr());
        System.out.println("本地访问机器的 IP:" + sr.getLocalAddr());
        if (sr.getRemoteHost().equals(sr.getLocalAddr())) {
            sr.setAttribute("login", "true");
        } else {
            sr.setAttribute("login", "false");
        }
    }
    public void requestDestroyed(ServletRequestEvent sre) {
```

```
            System.out.println("正在销毁");
        }
        public void attributeAdded(ServletRequestAttributeEvent srae) {
            System.out.print("request 范围内接收时  ---");
            System.out.println(srae.getName()+"="+srae.getValue());
        }
        public void attributeRemoved(ServletRequestAttributeEvent srae) {
            System.out.print("request 范围内取代另一个对象时    ---");
            System.out.println(srae.getName()+"="+srae.getValue());
        }
        public void attributeReplaced(ServletRequestAttributeEvent srae) {
            System.out.print("request 范围内移除时  ---");
            System.out.println(srae.getName()+"="+srae.getValue());}
}
```

（2）创建 index.jsp 页面，在页面的添加请求中加入 login 的值为 true 的对象，就可以直接进入 telnet.htm，如果不是 true 或者请求作用域中没有此对象，则显示当前页面，主要代码如下：

```
<%
String login=(String)request.getAttribute("login");
if(login.equals("true")){
response.sendRedirect("telnet.htm");
}
%>
<body><div align="center">
 <table width="335" height="225">
    <tr>
       <td bgcolor="lightblue" align="center">
   <form name="form1" method="post" action="telnet.htm">
  <div> 用户请登录:
    <input type="submit" name="Submit" value="登录">
    </div>
</form>
```

秘笈心法

对于请求监听，是在 Servlet 2.4 规范中新增的一个技术，当用户在监听程序中获取请求时就可以对请求进行统一处理，从而实现了监听服务器端免登录的功能。

第 8 章

JSTL 标签库

- ⏭ JSTL Core 标签库
- ⏭ JSTL I18N 标签库

8.1 JSTL Core 标签库

Core 标签库是 JSTL 的核心标签库，包括一般用途的标签、条件标签、迭代标签和 URL 相关的标签。下面通过几个实例介绍一些 Core 标签库的用法。

实例 195 利用 JSTL 标签实现网站计数器

初级

光盘位置：光盘\MR\08\195

实用指数：★★★

实例说明

本实例主要通过 JSTL 的<c:set>标签来实现“写入”的功能，自定义两个变量的作用域，分别定义为 application 和 session，运行结果如图 8.1 所示。



图 8.1　运行结果

关键技术

<c:set>标签的 value 值可以写在 value 属性中，也可以写在<c:set>标签体内，也就是说，set 标签还支持标签体，代码如下：

```
<c:set var="test" value="by property"></c:set>
<c:set var="test">by body</c:set>
```

<c:set>标签只有 5 个属性，分别为 var、value、scope、target 和 property。其中 var 是 set 的对象名，如果此对象不存在则自动生成，如果存在则将其修改，其中 scope 的作用范围有 session、request、page 和 application 4 种。

设计过程

创建 count.jsp 页面，使用<c:set>标签定义两个变量 allCount 和 count，其中 allCount 作用域为 application，而 count 作用域为 Session，并且页面被浏览累加两个变量实现计数器的功能，主要代码如下：

```
<table align="center" cellpadding="0" cellspacing="0" bgcolor="lightblue">
<c:set var="allCount" value="${ allCount + 1 }" scope="application"></c:set>
<c:set var="count" value="${ count + 1 }" scope="session"></c:set>
<Tr><td>
今天访问本网站总人数为：${ allCount } <br/>
今天您访问了此网站次数为：${ count } <br/>
</td></Tr>
<c:set var="test" value="by value property"></c:set>
<c:set var="test">by body</c:set>
</table>
<br/>
<br/>
<%
```

```
request.setAttribute("user", new com.mr.bean.User());
request.setAttribute("map", new java.util.HashMap());
%>
<c:set target="${ user }" property="name" value="${ param.name }"></c:set>
${ user.name }
<c:set target="${ map }" property="name" value="${ param.name }" />
${ map.name }
```

注意：set 标签只能设置 Interger Float Double 等数据类型，而不能直接操作 Java Bean 等复杂数据类型。

秘笈心法

target 作用和 var 有些相似，唯一不同的地方就是 target 只能操作 Java Bean 和 Map，它们的功能是相辅的，两者不能同时使用，target 只接受 EL 表达式，而 var 不能接受 EL 表达式。通常 target 一般与 property 一起使用，如果 target 为 Java Bean，则 property 为 Java Bean 的一个属性。

实例 196	根据参数请求显示到不同的页面 光盘位置：光盘\MR\08\196	初级 实用指数：★★★

实例说明

本实例将介绍如何应用<c:if>标签实现根据参数请求显示不同页面的功能。运行本实例，在页面中将根据<c:if>标签判断并显示出不同提示信息，如图 8.2 所示。



图 8.2　周二提示信息

关键技术

<c:if>标签能够实现 Java 语言中的 if 语句及 if-else 语句的功能，语法格式如下：

```
<c:if test="逻辑表达式"var="代表逻辑表达式的值的命名变量的名字"scope="(page|request|session|application) " />
```

<c:if>标签会把逻辑表达式的值存放在 var 属性指定的命名变量中，scope 属性则指定命名变量的范围，scope 属性的默认值是 page（页面范围）。

设计过程

创建 if.jsp 页面，把判断的条件（param.action==‘星期几？’）写在 test 的属性中，主要代码如下：

```
<fieldset>
		<c:if test="${ param.action == 'mon' }">
				周一了：工作的第一天，要加油哦
		</c:if>
		<c:if test="${ param.action == 'tues' }">
				周二了：工作了两天了，要适当补充体力哦
		</c:if>
		<c:if test="${ param.action == 'wed' }">
				周三了：忙碌的生活要学会调节
		</c:if>
		<c:if test="${ param.action == 'thu' }">
				周四了：偶尔偷下懒儿，不算过分哦
		</c:if>
		<c:if test="${ param.action == 'fri' }">
```

```
            周五了；加油明天就要休息了，HOHO
        </c:if>
        <c:if test="${ param.action == 'sat' }">
            周六了：和死党们出去 HAPPY 吧
        </c:if>
        <c:if test="${ param.action == 'sun' }">
            周日：要收敛一下活动，明个要上班呢
        </c:if>
</fieldset>
```

方法是根据 action 参数的值来达到不同星期的提示信息。action 为 sun（周日）时，在 URL 地址栏添加表达式“?action=sun”为周日的提示信息。

秘笈心法

<c:if>标签只是类似于单个的 if 语句，如果想实现类似 if-else 语句的功能，可以结合<c:choose>、<c:when>和<c:otherwise>标签使用。

实例 197　利用<c:forTokens>标签遍历字符串

初级

光盘位置：光盘\MR\08\197

实用指数：★★★

实例说明

本实例主要应用<c:forTokens>标签，实现遍历以特定分隔符分隔的字符串。运行本实例，使用<c:forTokens>循环标签，按“*”字符分隔其集合中所有的数据，如图 8.3 所示。

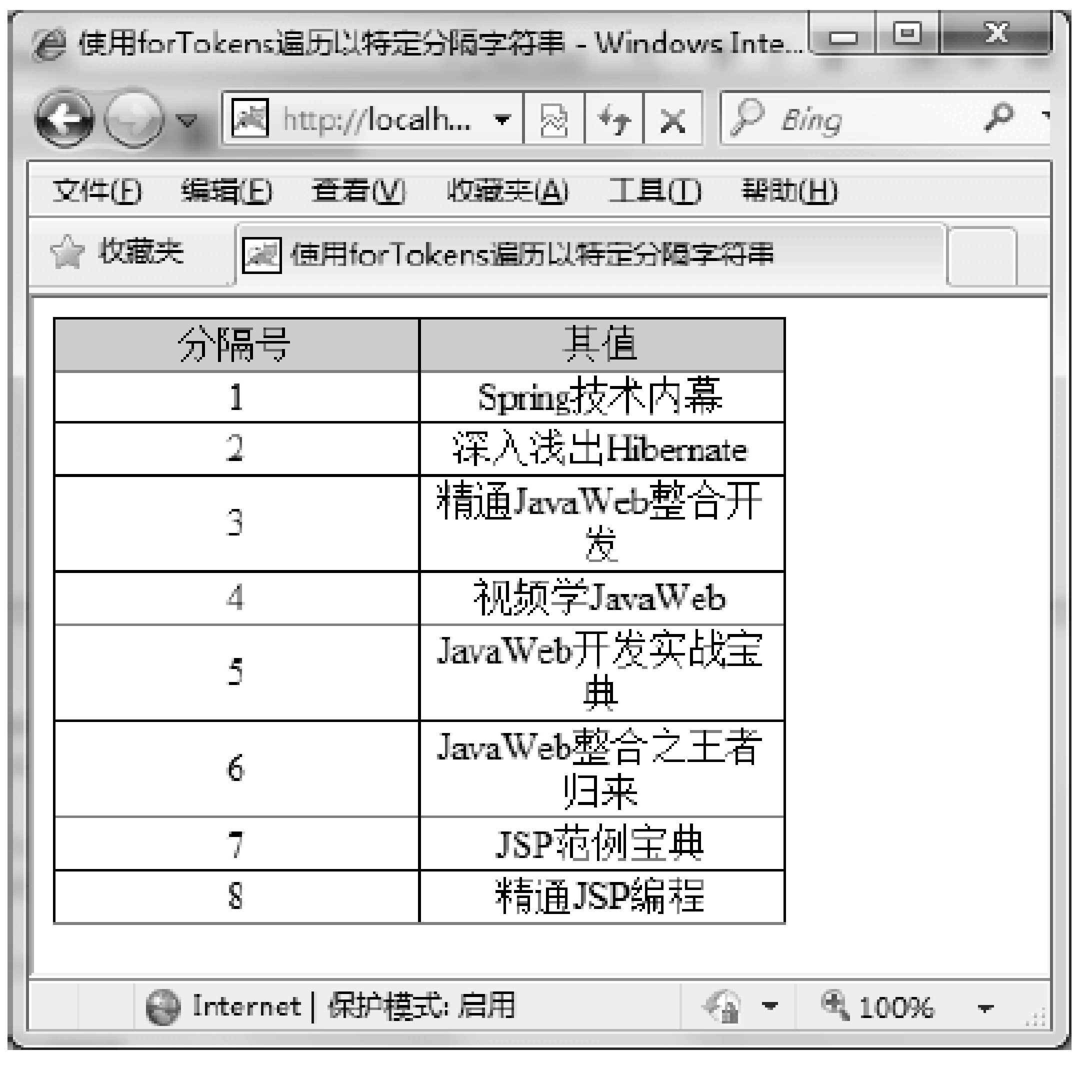


图 8.3　运行结果

关键技术

<c:forTokens>标签与<c:forEach>标签很相似，<c:forTokens>也有 begin、end、step、items 等属性，也可以遍历 itmes 属性的值。不同之处是，<c:forEach>标签中的 itmes 属性值为集合或数组，而<c:forTokens>标签中的 items 属性值是带分隔符的字符串。<c:forTokens>标签的属性如表 8.1 所示。

表 8.1　<c:forTokens>标签的属性

属　性	描　述
begin	返回 forTokens 标签 begin 属性的值
end	返回 forTokens 标签 end 属性的值
step	返回 forTokens 标签 step 属性的值
items	返回 forTokens 标签 items 属性的值

varStatus 为当前遍历对象的信息被记录在 varStatus 中，通过 varStatus 即可获取当前被遍历对象的信息。

设计过程

创建 forTokens.jsp 页面，<c:forTokens>标签的 items 属性有很多字符串，分别以“*”隔开，从第一个“*”字符开始分隔，直到第 8 个为止，主要代码如下：

```
<table>
  <tr bgcolor="#CCCCCC">
      <td>分隔号</td>
      <td>其值</td>
</tr>
      <c:forTokens
      items="JSP 开发王*Spring 技术内幕*深入浅出 Hibernate*精通 JavaWeb 整合开发*视频学 JavaWeb*JavaWeb 开发实战宝典*JavaWeb 整合之王者归来*JSP 范例宝典*精通 JSP 编程"
      delims="*" var="item" varStatus="varStatus" begin="1" end="8">
<tr>
      <td>${ varStatus.index }</td>
      <td>${ item }</td>
</tr>
      </c:forTokens>
</table>
```

秘笈心法

<c:forTokens>标签类似于 Java 中的 java.util.StringTokenizer 类的用法。

实例 198　利用 JSTL 选取随机数给予不同的提示信息

光盘位置：光盘\MR\08\198

初级

实用指数：★★★

实例说明

本实例将介绍如何利用 JSTL 标签实现选取随机数并给予不同的提示信息。运行本实例，效果如图 8.4 所示。

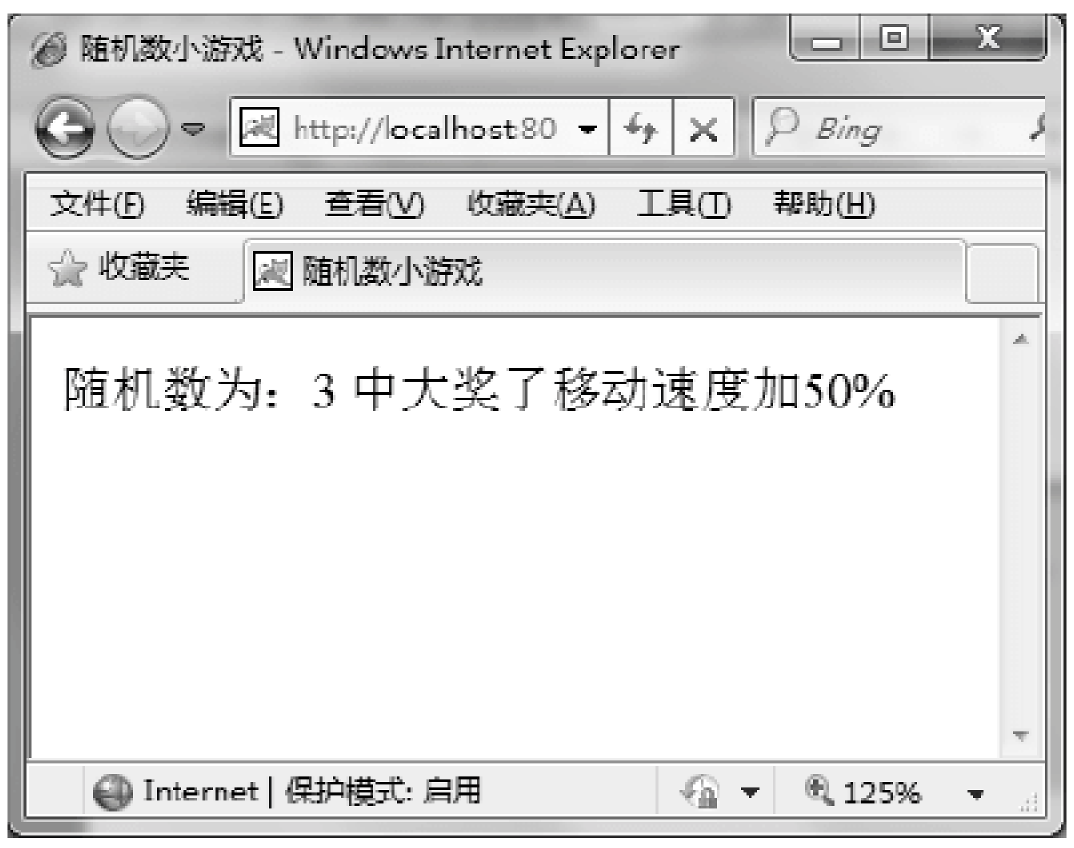


图 8.4　随机数为 3

关键技术

本实例主要通过应用<c:choose>、<c:when>和<c:otherwise>标签来实现。这几个标签在一起连用，可以实现 Java 语言的 if-else 语句的功能。

设计过程

创建 Random.jsp 页面，先将随机数保存在 num 变量中，然后再使用<c:choose>标签和<c:otherwise>标签根据随机数提示不同信息，主要代码如下：

```
  <body>
      <%Random rd = new Random(); %>
```

```
        <c:set var="num">
            <%=rd.nextInt(5)%>
        </c:set>
        <c:choose>
            <c:when test="${num==1}">随机数为：1 恭喜你前进 5 格</c:when>
            <c:when test="${num==2}">随机数为：2 十分晦气原地不动</c:when>
            <c:when test="${num==3}">随机数为：3 中大奖了移动速度加 50%</c:when>
            <c:when test="${num==4}">随机数为：4 舟车疲劳倒退 3 格</c:when>
            <c:otherwise>快去充电吧！！</c:otherwise>
        </c:choose>
    </body>
```

秘笈心法

使用<c:choose>、<c:when>和<c:otherwise>标签时，必须遵循以下语法规则：

- <c:when>和<c:otherwise>不能单独使用，它们必须位于<c:choose>父标签中。
- 在<c:choose>标签中可以包含一个或多个<c:when>标签。
- 在<c:choose>标签中可以不包含<c:otherwise>标签。
- 在<c:choose>标签中如果同时包含<c:when>和<c:otherwise>标签，那么<c:otherwise>标签必须位于<c:when>标签之后。

实例 199　利用<c:forEach>标签遍历 List 集合的元素

光盘位置：光盘\MR\08\199

初级

实用指数：★★★

实例说明

在 JSP 页面中，经常需要遍历保存数据的 List 集合，当然，可以应用 Java 脚本代码来实现，但是在 JSP 中插入大量的 Java 脚本是我们不希望看到的。针对这一问题，可以使用 JSTL 标签中的<c:forEach>标签。本实例将介绍如何应用<c:forEach>标签遍历 List 集合中的元素，运行结果如图 8.5 所示。

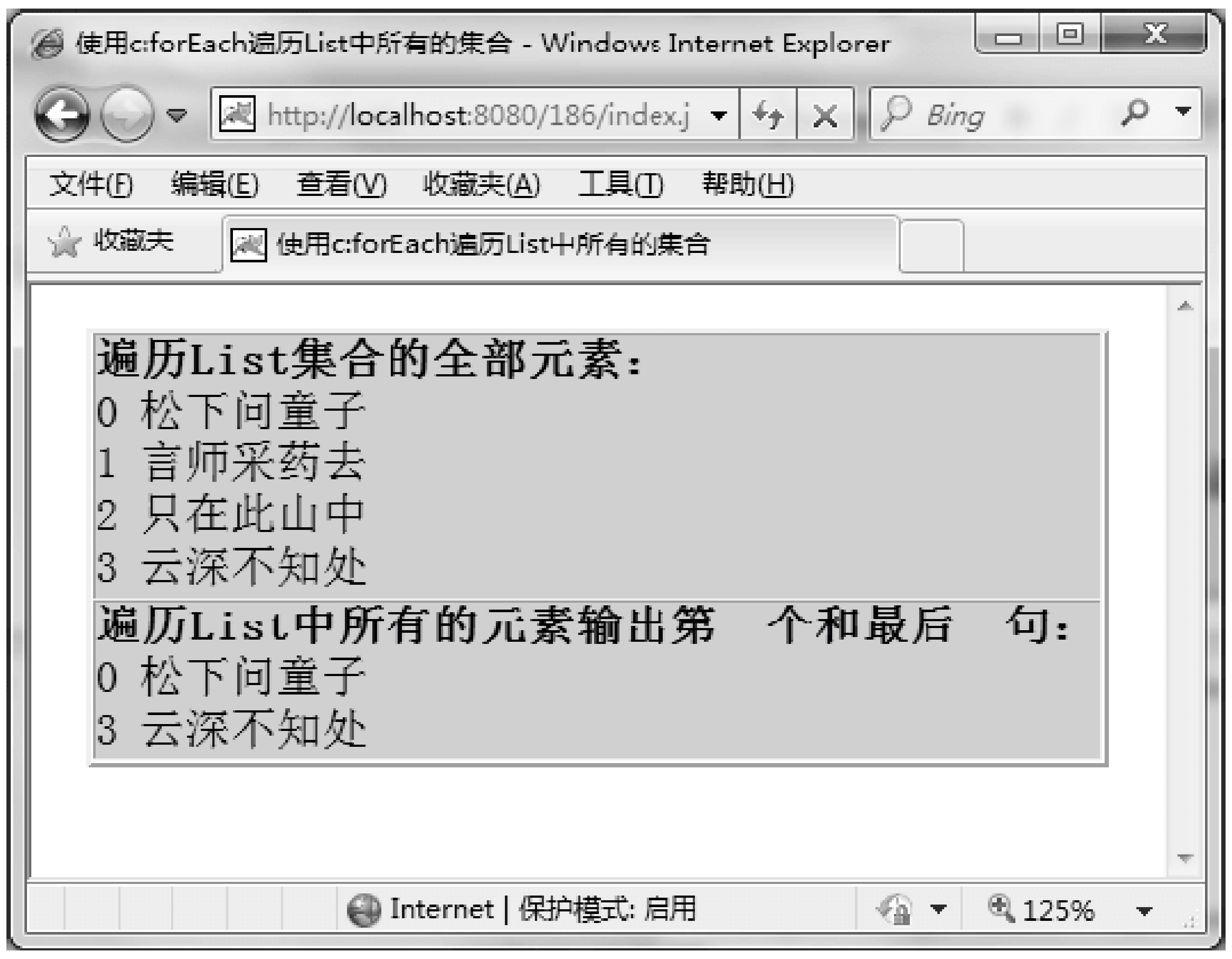


图 8.5　运行结果

关键技术

<c:forEach>标签用于遍历集合中的对象，并且能重复执行标签主体。<c:forEach>标签的基本语法如下：

```
<c:forEach var="代表集合中的一个元素的命名变量的名字" items="集合">
```

<c:forEach>标签每次从集合中取出一个元素，并且把它存放在 NESTED 范围内的命名变量中，在标签主体中可以访问这个命名变量。NESTED 范围是指当前标签主体构成的范围，只有当前标签主体才能够访问 NESTED 范围内的命名变量。

<c:forEach>标签的 varStatus 属性用于设置一个 javax.servlet.jsp.jstl.core.LoopTagStatus 类型的命名变量，它位于 NESTED 范围，这个命名变量包含了从集合中取出的当前元素的状态信息，如表 8.2 所示。

表 8.2　varStatus 属性的状态

属　性	描　述
index	当前元素在集合中的索引，从 0 开始计数
count	当前元素在集合中的序号，从 1 开始计数
first	当前元素是否为集合中的第一个元素
last	当前元素是否为集合中的最后一个元素

设计过程

创建 index.jsp 页面，首先声明一个 List 集合，其下有 4 个元素并保存在 request 作用域中。下面使用<c:forEach>标签遍历所有的元素，并且只筛选出第一条和最后一条元素，主要代码如下：

```
<body>
    <%
        List<String> list = new ArrayList<String>();    //创建 List 集合对象
        list.add("松下问童子");                          //添加 List 中的元素
        list.add("言师采药去");
        list.add("只在此山中");
        list.add("云深不知处");
        request.setAttribute("list", list);            //将 List 集合保存到 request 对象中
    %>
<table align="center" cellpadding="0" cellspacing="0" border="2"
        bgcolor="lightblue">
    <Tr>
        <Td><b>遍历 List 集合的全部元素：</b><br>
    <c:forEach items="${requestScope.list}" var="keyvalue"
            varStatus="id">
                ${id.index } ${keyvalue}<br>
    </c:forEach>
        </Td>
    </Tr>
    <Tr>
        <td><b>遍历 List 中所有的元素输出第一个和最后一句：</b><br>
<c:forEach items="${requestScope.list}" var="keyvalue"
        varStatus="id" begin="0" step="3">
            ${id.index } ${keyvalue}<br>
</c:forEach>
        </td>
    </Tr>
</table>
```

秘笈心法

当使用 forEach 遍历集合，既不知道遍历对象是所有对象中的第几个，也不知道谁是第一个和最后一个时，想知道类似的信息就需要借助 forEach 的 varStatus 属性，此属性为 javax.servlet.jsp.jstl.core.LoopTagStatus 类的对象。

实例 200	利用 JSTL 标签导入用户注册协议 光盘位置：光盘\MR\08\200	初级 实用指数：★★★

实例说明

在网站开发过程中，在用户注册或者其他注册时，在注册之前，需要展示给用户注册协议信息，只有同意

了协议才可以继续完成注册。本实例将应用 JSTL 标签中的<c:import>标签，实现导入注册协议的文本资源文件，运行结果如图 8.6 所示。

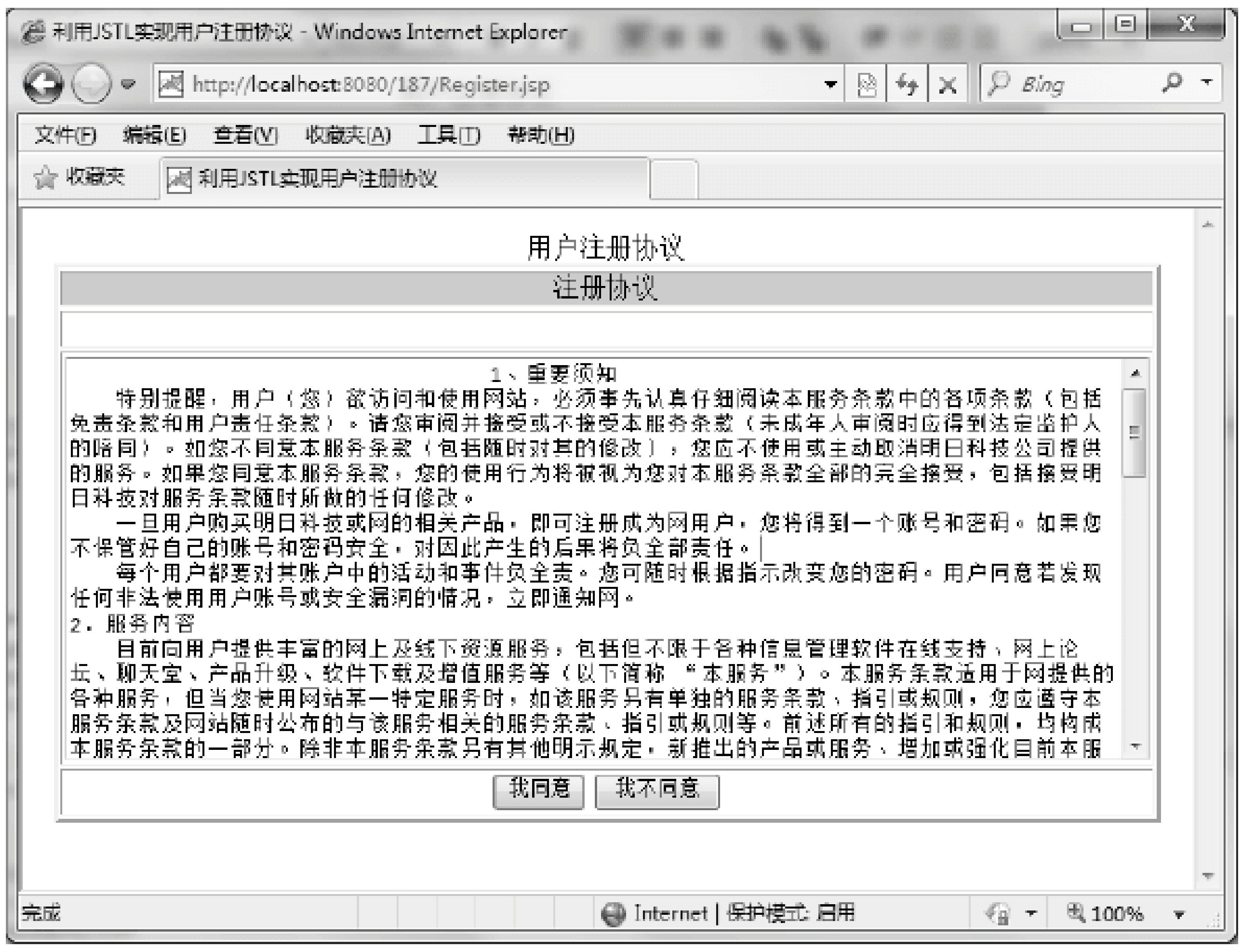


图 8.6　用户注册协议

关键技术

<c:import>标签用于包含其他 Web 资源，它与<jsp:include>指令的作用有些类似。<c:import>标签与<jsp:include>标签的区别在于，前者不仅可以包含同一个 Web 应用中的资源，还可以包含其他 Web 应用中的资源，甚至是其他网站的资源。<c:import>标签的基本语法格式如下：

```
<c:import url="Web 资源的 URL">
```

<c:import>标签的属性如表 8.3 所示。

表 8.3　<c:import>标签的属性

属　　性	描　　述	EL
url	要导入的文件资源的 URL 路径或地址	可以
context	要访问到同一个服务器的其他 Web 工程，其值必须以“/”开头，如果指定了该属性，其属性值也必须以“/”开头	可以
var	变量名称可以将获取的资源存储在变量中	不可以
scope	变量的作用域	不可以
varReader	以 Reader 类型存储要包含文件的内容	不可以
charEncoding	要导入文件的编码格式	可以

设计过程

创建 Register.jsp 页面，使用<c:import>的 url 属性把注册协议的文本资源文件 agreement.txt 导入到页面中的文本域，关键代码如下：

```
<body><div align="center">用户注册协议</div>
  <table align="center" border="2">
      <tr bgcolor="#CCCCCC">
            <Td align="center">注册协议</Td>
      </tr>
      <Tr>
            <td> </td>
      </Tr>
      <tr>
            <Td><textarea rows="15" cols="80">
                  <c:import url="agreement.txt" charEncoding="gbk"/>
```

```
            </textarea></Td>
        </tr>
        <Tr>
            <td align="center" colspan="2"><input type="submit" value="我同意"/>
            <input type="submit" value="我不同意"/></td>
            </Tr>
        </table>
</body>
```

秘笈心法

类似于 JSP 的<%file include%>与<jsp:include>，JSTL 也提供了实现 include 功能的标签<c:import>，不过 JSTL 标签功能强大，可以把 Internet 的网页包含进来，例如：

```
<c:import url="http://www.sina.com.cn" charEncoding="gb2312"></c:import>
```

8.2　JSTL I18N 标签库

I18N 是 Internationalization 的简称，因为该单词的首字母 I 与尾字母 n 中间隔着 18 个字符，由此得名。I18N 标签库主要用于编写国际化的 Web 应用。I18N 标签库中的标签可以分为两部分：一部分用于国际化，另一部分用于对时间、日期和数字进行格式化。

实例 201　利用 JSTL 标签设置请求的字符编码　　初级

光盘位置：光盘\MR\08\201　　实用指数：★★★

实例说明

本实例将介绍如何应用 JSTL 标签设置请求的字符编码。运行本实例，在页面中单击“验证”按钮，转换预设的编码格式，运行结果如图 8.7 所示。

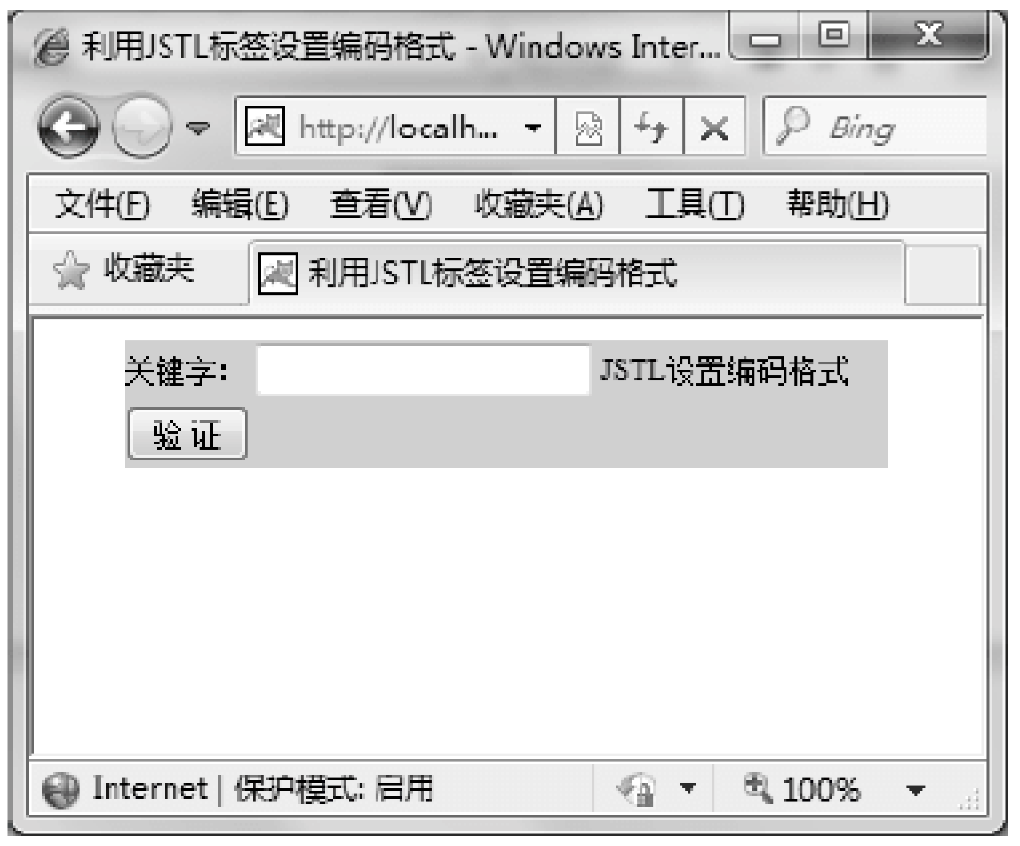


图 8.7　运行结果

关键技术

本实例主要应用 JSTL 标签库的 I18N 标签中的<fmt:requestEncoding>标签，该标签用于设置 HTTP 请求正文使用的字符编码，基本语法如下：

```
<fmt:requestEncoding  value="GB2312">
```

该标签设置的请求编码，等价于应用 request 对象的 setCharacterEncoding()方法设置的请求编码，代码如下：

```
<%  request.setCharacterEncoding("GB2312")%>
```

设计过程

创建 Encoding.jsp 页面，使用 requestEncoding 的 value 设置编码格式为 UTF-8，主要代码如下：

```
<fmt:requestEncoding value="UTF-8" />
<form action="${ pageContext.request.requestURI }" method="post">
        <table height="50" width="300" cellpadding="0" cellspacing="0"
                align="center" bgcolor="lightblue">
                <Tr>
                        <Td>关键字：<input name="key" />
        <c:out value="${ param.key }" default="请输入"></c:out>
                        </Td>
                </Tr>
                <tr>
                        <Td><input type="submit" value="验  证">
                        </Td>
                </tr>
        </table>
</form>
```

如果去掉 requestEncoding 标签，运行时就会出现乱码。

秘笈心法

如果使用 get 方法提交，还要修改 tomcat 的 server.xml，将 URIEncoding 设置为 UTF-8，否则一样会出现乱码。

实例 202 利用 JSTL 标签实现国际化

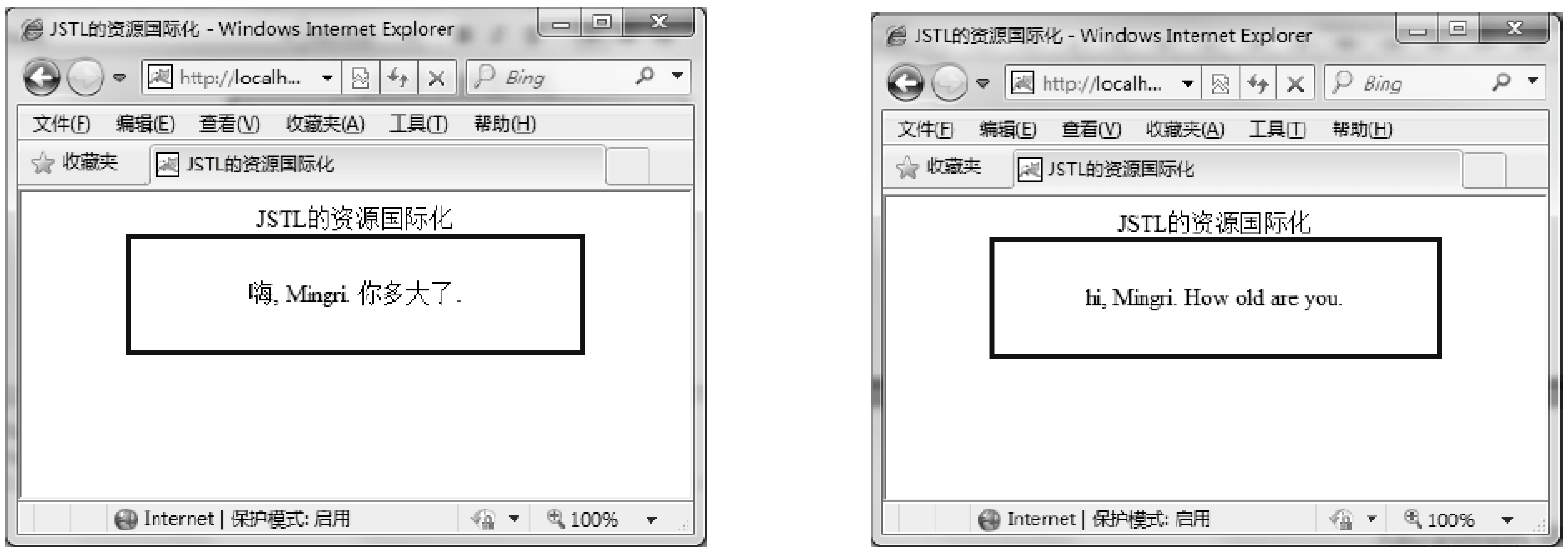
光盘位置：光盘\MR\08\202

初级

实用指数：★★★

实例说明

本实例使用<fmt:bundle>、<fmt:message>和<fmt:param>标签来实现资源国际化，把所有中文的提示信息放在 message_zh_CN.properties 文件中，所有英文的放在 message_en_US.properties 文件中，系统以哪种语言登录时，就以哪种语言显示，运行结果如图 8.8 所示。



（a）　　　　（b）

图 8.8　运行结果

关键技术

❑ <fmt:bundle>标签

该标签用于设置标签主要使用的 ResourceBundle，基本语法如下：

```
<fmt:bundle basename=""前缀="消息 key 的前缀">
标签主体
</fmt:bundle>
```

❑ <fmt:message>标签

该标签根据属性 key 返回 ResourceBundle 中匹配的消息文本。假定在 message.properties 资源文件中包含如下内容：

```
myword=message    from message.properties
```

然后在 JSP 中可以应用<fmt:message>标签从资源文件中读取消息文本，代码如下：

```
<fmt:bundle basename="resource">
        <fmt:message key=" myword" /><br>
</fmt:bundle>
```

❑ <fmt:param>标签

该标签嵌套在<fmt:message>父标签中，用于为消息文本中的消息参数设置值。假定在 message.properties 文件中包含如下内容：

```
username=My name is {0},I'm {1} yesrs old.
```

在消息文本中的{0}和{1}分别表示两个消息参数。在 JSP 中，以下代码演示了<fmt:param>标签的用法：

```
……
<fmt:message key="username">
        <fmt:param value=" John" />
        <fmt:param value="nineteen">
</fmt:message>
……
```

以上代码中的两个<fmt:param>标签用于替换消息文本中的两个参数，输出结果如下：

```
My my is John,I'm nineteen years old.
```

设计过程

（1）首先创建 message_zh_CN.properties 中文资源文件和 message_en_US.properties 英文资源文件，并存放在 src 包下，其内容如下所示：

message_zh_CN.properties：

```
prompt.hi = \u55E8, {0}.
prompt.greeting =\u4f60\u591a\u5927\u4e86.
```

message_en_US.properties：

```
prompt.hi = hi, {0}.
prompt.greeting =How old are you.
```

（2）创建 fmt.jsp 页面文件，这里引用<fmt:bundle>、<fmt:message>和<fmt:param>标签，主要代码如下：

```
<div align="center">JSTL 的资源国际化</div>
<table align="center" cellpadding="0" cellspacing="0" height="80" width="300" border="2" bordercolor="blue">
<fmt:bundle basename="messages">
<Tr>
<Td align="center">
 <fmt:message key="prompt.hi">
        <fmt:param value="Mingri"></fmt:param>
 </fmt:message>
 <fmt:message key="prompt.greeting"></fmt:message></Td>
</Tr>
</fmt:bundle>
</table>
```

秘笈心法

<fmt:message>标签有一个 var 属性和 scope 属性，用于把消息文本保存在特定范围内。var 属性指定命名变量的名字；scope 属性指定消息文本的存放范围，该属性的默认值为 page。如果没有设定 var 和 scope 属性，那么<fmt:message>标签直接输出消息文本，否则将把消息文本作为命名变量存放在特定范围内。

实例 203	利用<fmt:setLocale>显示所有地区的数据格式 光盘位置：光盘\MR\08\203	初级 实用指数：★★★

实例说明

如果一个应用程序被跨地区的用户使用，那么肯定会有地区间的文化差异，本实例使用<fmt:setLocale>标签设置本地化的日期格式、货币格式来解决本地化的问题，运行结果如图 8.9 所示。



图 8.9　运行结果

关键技术

<fmt:setLocale>标签用于设置 Locale，把 Locale 保存在特定范围内，其基本语法如下：

```
<fmt:setLocale value="Local"    scope="{page|request|session|application}" />
```

<fmt:setLocale>标签的 scope 属性默认值为 page。以下代码在会话范围内存放了一个表示中文的 Locale：

```
<fmt:selLocal value="zh_CN" scope="session">
```

以上<fmt:setLocale>标签和以下 Java 代码片段的作用是等价的：

```
<%
        Locale locale=new Locale("zh", " CN");
        Session.setAttribute("javax.servlet.jsp.jstl.fmt.locale.session",locale);
%>
```

提示： Locale（本地）指的是一个具有相同风俗、文化和语言的区域。如果一个应用没有事先把 I18N 作为内嵌的功能，那么当这个应用需要支持新的 Locale 时，开发人员必须将嵌入在源代码中的文本、图片和消息进行修改，然后重新编译源代码。Locale 类的实例代表一种特定的语言和地区。如果 Java 类库中某个类在运行时需要根据 Locale 对象来调整其功能，那么就策划这个类是本地敏感的（Locale-Sensitive）。

设计过程

创建 locale.jsp 页面，使用<fmt:setLocale>标签输出 Locale，再把所有的 Locale 存储到 request 作用域中，利用 JSTL 的<c:forEach>标签循环迭代出所有的 Locale 的日期、数字、货币格式，主要代码如下：

```
<body>
<%                    //把所有的 Locale 存储到 request 作用域中
request.setAttribute("localeList", Locale.getAvailableLocales());
%>
<div>
本地区为:<%=Locale.getDefault().getDisplayName()%>
</div>
<table>
<tr class="title">
        <th>地区</th>
        <th>语言</th>
        <th>日期时间格式</th>
        <th>数字格式</th>
        <th>货币格式</th>
</tr>
<jsp:useBean id="date" class="java.util.Date"></jsp:useBean>
<c:forEach var="locale" items="${ localeList }">
        <fmt:setLocale value="${ locale }" />
        <tr>
```

```
                <td align="left">${ locale.displayName }</td>
                <td align="left">${ locale.displayLanguage }</td>
                <td><fmt:formatDate value="${ date }" type="both"/></td>
                <td><fmt:formatNumber value="100900.9" /></td>
                <td><fmt:formatNumber value="100900.9" type="currency" /> </td>
        </tr>
</c:forEach>
</table>
```

秘笈心法

需要注意的是，中文的资源文件中的中文信息，需要进行字符编码转换，转换为“\uxxxx”模式的，需要利用 JDK 提供的 native2ascii.exe 可执行程序进行转换。在 DOS 下执行以下命令，将生成按 GB2312 编码的中文资源文件 message_zh_CN.properties：

```
native2ascii –encoding gb2312 message_temp.properties message_zh_CN.properties
```

以上命令生成的 messsage_zh_CN.properties 文件的内容如下：

```
……
login.user = \u7528\u6237\u540d
login.pwd=\u53e3\u4ee4
……
```

实例 204　利用<fmt:timeZone>显示不同地区的时间

初级

光盘位置：光盘\MR\08\204

实用指数：★★★

实例说明

本实例使用<fmt:timeZone>标签获取所有地区时区并保存在 map()对象中，再使用<c:forEach>迭代标签遍历所有的地区时间，运行结果如图 8.10 所示。



时区编号	时区	当地时间	本地时差
Etc/GMT+12	GMT-12:00	2010-7-15 15:29:12	-12.0
Etc/GMT+11	GMT-11:00	2010-7-15 16:29:12	-11.0
MIT	西萨摩亚时间	2010-7-15 16:29:12	-11.0
Pacific/Apia	西萨摩亚时间	2010-7-15 16:29:12	-11.0
Pacific/Midway	萨摩亚群岛标准时间	2010-7-15 16:29:12	-11.0
Pacific/Niue	纽威岛时间	2010-7-15 16:29:12	-11.0
Pacific/Pago_Pago	萨摩亚群岛标准时间	2010-7-15 16:29:12	-11.0
Pacific/Samoa	萨摩亚群岛标准时间	2010-7-15 16:29:12	-11.0
US/Samoa	萨摩亚群岛标准时间	2010-7-15 16:29:12	-11.0
America/Adak	夏威夷-阿留申群岛标准时间	2010-7-15 18:29:12	-10.0
America/Atka	夏威夷-阿留申群岛标准时间	2010-7-15 18:29:12	-10.0
Etc/GMT+10	GMT-10:00	2010-7-15 17:29:12	-10.0
HST	夏威夷标准时间	2010-7-15 17:29:12	-10.0
Pacific/Fakaofo	托克劳群岛时间	2010-7-15 17:29:12	-10.0
Pacific/Honolulu	夏威夷标准时间	2010-7-15 17:29:12	-10.0
Pacific/Johnston	夏威夷标准时间	2010-7-15 17:29:12	-10.0

图 8.10　运行结果

关键技术

<fmt:timeZone>标签用于设置当前标签主体使用的时区，其基本语法如下：

```
<fmt:timeZone value="时区">
        标签主体
</fmt:timeZone>
```

例如，以下代码输出采用 GMT 时区的日期和时间：

```
<fmt:timeZone value="GMT">
        <fmt:formatDate value="<%=new Date()%>" type=" both"/>
</fmt:timeZone>
```

假定当前上下文使用的 Locale 为 en_US，以上代码的输出结果为：

```
Aug 17,2010 2:15:48 AM
```

设计过程

创建 time.jsp 页面文件，声明一个 Map 对象，并使用 for 循环把所有的地区时间保存在 Map 对象中，并存储到 request 范围内，然后用<fmt:timeZone>标签获取地区时间，最后使用<c:forEach>把所有的数据显示在页面中，主要代码如下：

```
<div align="center">
<%
Map<String, TimeZone> hashMap = new HashMap<String, TimeZone>();
for(String ID : TimeZone.getAvailableIDs()){
        hashMap.put(ID, TimeZone.getTimeZone(ID));
}
request.setAttribute("timeZoneIds", TimeZone.getAvailableIDs());
request.setAttribute("timeZone", hashMap);
%>
<jsp:useBean id="date" class="java.util.Date"></jsp:useBean>
<fmt:setLocale value="zh_CN" />
北京时间：<%= TimeZone.getDefault().getDisplayName() %>
<fmt:formatDate value="${ date }" type="both" /> <br/>
</div>
<table align="center">
<tr>
        <th>时区编号</th>
        <th>时区</th>
        <th>当地时间</th>
        <th>本地时差</th>
</tr>
<c:forEach var="ID" items="${ timeZoneIds }" varStatus="status">
<tr>
        <td>${ ID }</td>
        <td>${ timeZone[ID].displayName }</td>
        <td>
                <fmt:timeZone value="${ ID }" >
                        <fmt:formatDate value="${ date }" type="both" timeZone="${ ID }"/>
                </fmt:timeZone>
        </td>
        <td>${ timeZone[ID].rawOffset / 60 / 60 / 1000 }</td>
</tr>
</c:forEach>
</table>
```

秘笈心法

在 JSTL 标签的 I18N 标签库中，还包含一个<fmt:setTimeZone>标签，用于设置时区，把时区保存到特定范围内，其基本语法如下：

```
<fmt:setTimeZone value="时区" var="命名变量的名字" scope="{page|request|session|application}" />
```

实例 205	利用<fmt:formatDate>标签对日期格式化 光盘位置：光盘\MR\08\205	初级 实用指数：★★★

实例说明

本实例使用<fmt:formatDate>标签对日期格式化，将实现对所有地区的日期时间进行转换，运行结果如图8.11所示。



日期格式化标签

地区缩写	日期与时间	数字	货币
en	Jul 16, 2010 11:31:22 AM	430,000.52	¤430,000.52
fr	16 juil. 2010 11:31:22	430 000,52	430 000,52 ¤
de	16.07.2010 11:31:22	430.000,52	¤ 430.000,52
it	16-lug-2010 11.31.22	430.000,52	¤ 430.000,52
ja	2010/07/16 11:31:22	430,000.52	¤ 430,000.52
ko	2010. 7. 16 오전 11:31:22	430,000.52	¤ 430,000.52
zh	2010-7-16 11:31:22	430,000.52	¤ 430,000.52
zh_CN	2010-7-16 11:31:22	430,000.52	￥430,000.52
zh_TW	2010/7/16 上午 11:31:22	430,000.52	NT$430,000.52
fr_FR	16 juil. 2010 11:31:22	430 000,52	430 000,52 €
de_DE	16.07.2010 11:31:22	430.000,52	430.000,52 €
it_IT	16-lug-2010 11.31.22	430.000,52	€ 430.000,52
ja_JP	2010/07/16 11:31:22	430,000.52	￥430,001
ko_KR	2010. 7. 16 오전 11:31:22	430,000.52	₩430,001
zh_CN	2010-7-16 11:31:22	430,000.52	￥430,000.52

图 8.11　运行结果

关键技术

<fmt:formatDate>标签用于对日期和时间进行格式化，它的属性介绍如下。

❶ value 属性：需要格式化的日期或时间。

❷ type 属性：指定格式化日期或时间或日期和时间。可选值包括 date、time 和 both。默认值为 date。

❸ dateStyle 属性：日期的格式化样式。默认值为 default。

❹ timeStyle 属性：时间的格式化样式。默认值为 default。

❺ pattern 属性：指定自定义的格式化日期和时间的样式。

❻ timeZone 属性：指定使用的时区。

❼ var 属性：指定命名变量的名字。

❽ scope 属性：指定经过格式化的时间和日期的存放范围。默认值为 page。

设计过程

创建 DataFormat.jsp 页面，首先获取所有的 field 和 Locale，并进行遍历所有的 field，如果类型为 Locale，则将其添加到 localeList 中，创建一个 double[]数组保存到 request 中。使用<fmt:formatDate>标签的 value 属性设置要格式化的日期，最后用<c:forEach>把所有信息迭代到页面中，主要代码如下：

```
<div align="center">日期格式化标签</div>
<%
 Field[] field = Locale.class.getFields();
 List<Locale> localeList = new ArrayList<Locale>();
 for(int i=0; i<field.length; i++){
       if(field[i].getType().equals(Locale.class)){
             localeList.add((Locale)field[i].get(null));
       }
 }
 request.setAttribute("localeList", localeList);
%>
<table align="center">
<tr>
        <td>地区缩写</td>
        <td>日期与时间</td>
        <td>数字</td>
        <td>货币</td>
</tr>
<jsp:useBean id="date" class="java.util.Date"></jsp:useBean>
<c:forEach var="locale" items="${ localeList }">
        <fmt:setLocale value="${ locale }"/>
```

```
        <tr>
                <td>${ locale }</td>
                <td><fmt:formatDate value="${ date }" type="both"/></td>
                <td><fmt:formatNumber value="430000.52" /></td>
                <td><fmt:formatNumber value="430000.52" type="currency" /> </td>
        </tr>
</c:forEach>
</table>
```

秘笈心法

对于<fmt:formatDate>标签的 dateStyle 属性和 timeStyle 属性的详细用法，可参考 java.text.DateFormat 类的 API 文档；对于 pattern 属性的用法，可参考 java.text.SimpleDateFormat 类的 API 文档。

第 9 章

JavaScript 技术

- 数据验证
- 字符串处理
- 日期时间处理
- 使用 JavaScript 控制 DOM

9.1 数 据 验 证

在 Web 应用开发过程中，为了保证提交的表单数据的有效性和准确性，需要对表单数据进行验证。本节将介绍一些常用的数据验证的 JavaScript 函数。

实例 206 通过正则表达式验证日期

光盘位置：光盘\MR\09\206

初级

实用指数：★★★☆

实例说明

本实例将介绍在 JavaScript 中，如何利用正则表达式来验证表单中输入的日期类型的值是否有效，运行结果如图 9.1 所示。

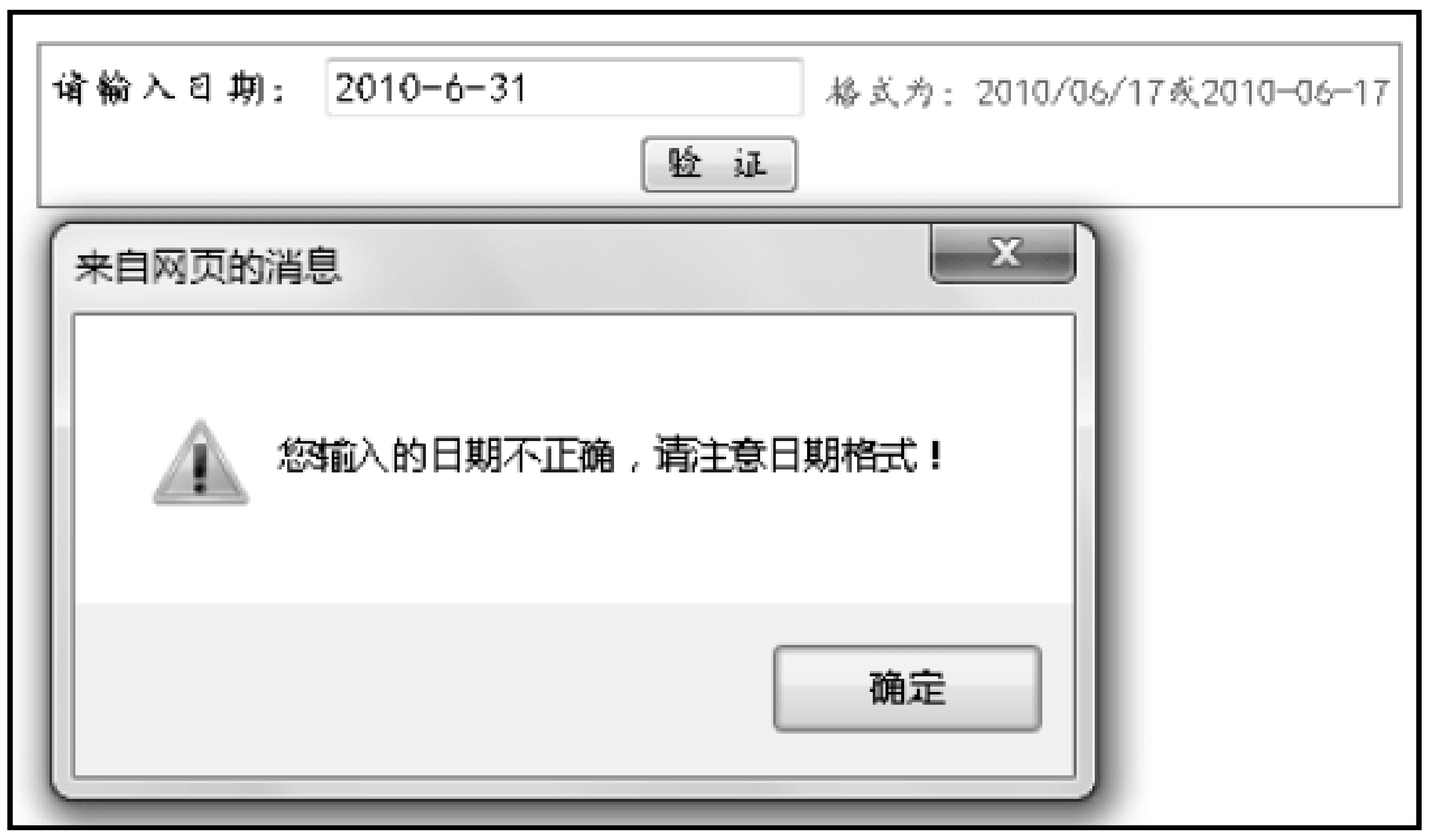


图 9.1 使用正则表达式验证日期

关键技术

在 JavaScript 中使用正则表达式主要是通过正则表达式对象实现的，使用正则表达式的语法格式如下：

```
re = /pattern/[flags]
```

参数说明

❶ re：必选参数。将要赋值为正则表达式模式的变量名。

❷ pattern：必选参数。要使用的正则表达式模式，以“/”开头和结尾。

❸ flags：可选参数。用于指定匹配的标记，常用的标记有 g（全文查找出现的所有 pattern）、I（忽略大小写）和 m（多行查找）。

验证日期格式是否正确时，需要注意以下几点：

（1）年份由 4 位数字组成。

（2）月份由 1～12 的数字组成。

（3）第 1、3、5、7、8、10、12 月份为 31 天，4、6、9、11 月份为 30 天。

（4）闰年的 2 月份为 29 天，非闰年的 2 月份为 28 天。

设计过程

（1）新建 index.jsp 页，在该页的<script>标签中利用正则表达式编写验证日期的 JavaScript 函数，如果验证成功返回 true，否则返回 false，关键代码如下：

```
function CheckDate(str){
	var Expression = /^((((1[6-9]|[2-9]\d)\d{2})(\/|\-)(0?[13578]|1[02])(\/|\-)(0?[1-9]|[12]\d|3[01]))|
(((1[6-9]|[2-9]\d)\d{2})(\/|\-)(0?[13456789]|1[012])(\/|\-)(0?[1-9]|[12]\d|30))|(((1[6-9]|[2-9]\d)\d{2})(\/|\-)0?2(\/|\-)(0?[1-9]|1\d|2[0-8]))|(((1[6-9]|[2-9]\d)(0
[48]|[2468][048]|[13579][26])|((16|[2468][048]|[3579][26])00))-0?2-29-))$/;
```

```
        if(Expression.test(str)){
                return true;
        }else{
                return false;
        }
}
```

（2） 编写按钮的 onClick 事件调用的 JavaScript 方法，在方法中验证用户输入的日期是否合法，如果验证失败则弹出提示信息，否则提交表单，关键代码如下：

```
function check(){
        var mydate = document.getElementById("mydate");    //获得日期文本框对象
        if(mydate.value==""){                               //判断输入的日期是否为空
                alert("请输入日期！");
                mydate.focus();                             //使文本框获得焦点
                return;
        }
        if(!CheckDate(mydate.value)){                       //验证日期格式是否正确
                alert("您输入的日期不正确，请注意日期格式！");
                mydate.focus();
                return;
        }
        document.getElementById("dataForm").submit();
        document.forms.dataForm.submit();
}
```

秘笈心法

由于验证日期的正则表达式比较繁琐，并且不容易理解，因此笔者不建议用此方法来验证日期是否合法。

实例 207	验证输入的日期是否正确 光盘位置：光盘\MR\09\207	初级 实用指数：★★★★

实例说明

实例 206 已经介绍了如何通过正则表达式来验证日期的合法性，验证日期的正则表达式编写时很繁琐，且很容易出现错误。本实例将介绍如何通过普通的 JavaScript 业务逻辑方法来验证日期的合法性，运行结果如图 9.2 所示。

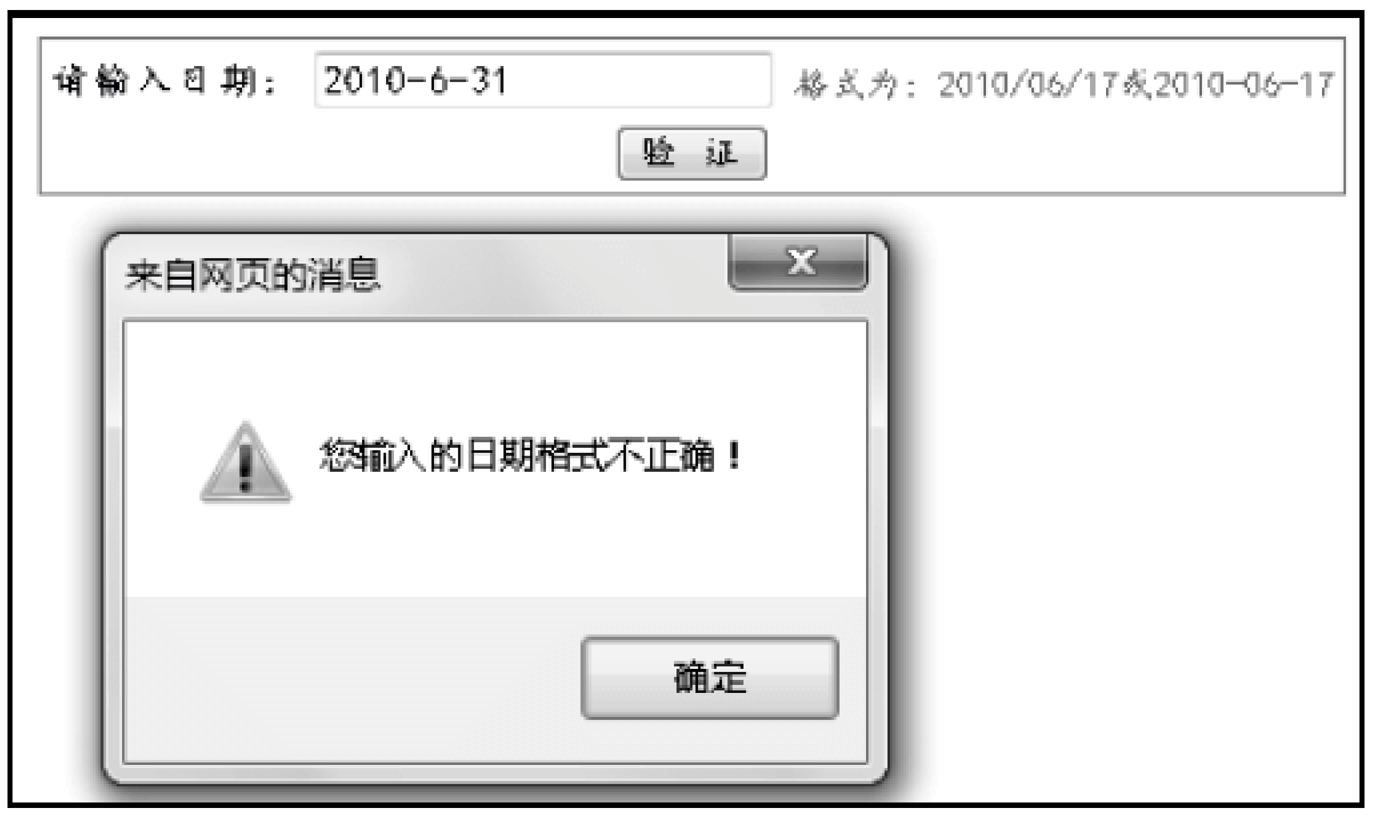


图 9.2　验证输入的日期是否合法

关键技术

应用 JavaScript 编写验证输入的日期是否合法时，需要注意以下几点：

（1）日期中的年份、月份、日都必须是大于 0 的数字。

（2） 判断输入的年份是否为闰年，并判断 2 月份的天数，闰年的 2 月份为 29 天，非闰年的 2 月份为 28 天。

（3）1、3、5、7、8、10、12 月份为 31 天，4、6、9、11 月份为 30 天。

设计过程

（1）新建 index.jsp 页，在该页的<script>标签中编写验证日期是否合法的 JavaScript 方法，关键代码如下：

```
function checkDate(dateStr){
	if(dateStr==""||dateStr==null){
		return false;
	}
	else{
		if(dateStr.indexOf("-")!=-1){
			var dateArr = dateStr.split("-");
			var year = dateArr[0];		//提取年份
			var month = dateArr[1];	//提取月份
			var day = dateArr[2];		//提取日
			//如果年份、月份、日期不是数字或者<=0，则返回 false
			if(isNaN(year)||year<=0){
				return false;
			}
			if(isNaN(month)||month<=0||month>12){
				return false;
			}
			if(isNaN(day)||day<=0||day>31){
				return false;
			}
			//年份能被 4 整除并且不能被 100 整除，或者能被 400 整除，则为闰年
			if((year%4==0&&year%100!=0)||(year%400==0)){
				if(month==2){//闰年的 2 月
					if(day>29){
						return false;
					}
				}
			}else{//不是闰年的 2 月
				if(month==2){
					if(day>28){
						return false;
					}
				}
			}
			//1、3、5、7、8、10、12 月份为 31 天
			var m1 = new Array(1,3,5,7,8,10,12);
			for(var i=0;i<m1.length;i++){
				if(parseInt(month)==m1[i]){
					if(day>31){
						return false;
					}
				}
			}
			//4、6、9、11 月份为 30 天
			var m2 = new Array(4,6,9,11);
			for(var j=0;j<m2.length;j++){
				if(parseInt(month)==m2[j]){
					if(day>30){
						return false;
					}
				}
			}
		}else{
			return false;
		}
	}
	return true;
}
```

（2）编写表单提交按钮的 onClick 事件调用的 JavaScript 方法，在该方法中调用验证日期的方法 checkDate()，验证成功则提交表单，否则弹出验证失败的提示信息，关键代码如下：

```
function check(){
	var mydate = document.getElementById("mydate");
	if(checkDate(mydate.value)){
```

```
				document.getElementById("dateForm").submit();
		}else{
				if(mydate.value==""||mydate.value==null){
					alert("请输入日期！");
					mydate.focus();		//日期文本框获得焦点
					return ;
				}
				mydate.focus();
				alert("您输入的日期格式不正确！");
		}
}
```

秘笈心法

根据本实例，读者可以在会员或用户注册页面中判断输入的生日是否正确，也可以在日期型数据查询页面中判断输入的日期是否正确。

实例 208　检查表单元素的值是否为空

光盘位置：光盘\MR\09\208

初级

实用指数：★★★★

实例说明

在实际的开发过程中，经常需要判断用户提交的表单中某个元素的值是否为空，还有一种情况是表单中所有元素的值都不允许为空。本实例将介绍一种简单有效的判断表单中所有元素是否为空的方法。运行本实例，如图 9.3 所示，如果在表单中没有输入信息，单击“提交”按钮将弹出提示信息。



图 9.3　检查表单元素的值是否为空

关键技术

本实例主要是在 JavaScript 中通过循环 form 对象的 elements 属性来实现的。form 对象的 elements 属性也就是页面中 form 表单的所有元素的数组，例如，form.elements[0]表示表单第一个元素对象，form.elements[n]表示表单第 n 个元素对象。

设计过程

（1）新建 index.jsp 表单页，该页的表单中包含 3 个不允许为空的元素和一个“提交”按钮，并且需要定义一个表单的 id 属性值，关键代码如下：

```
<form action="" id="myform">
	<table align="center">
		<tr>
			<td>留言人：</td>
			<td>
				<input type="text" name="messageUser" title="留言人">
```

```
                </td>
            </tr>
            <tr>
                <td>留言标题：</td>
                <td>
                    <input type="text" name="messageTitle" title="留言标题">
                </td>
            </tr>
            <tr>
                <td>留言内容：</td>
                <td>
                    <textarea rows="8" cols="45" title="留言内容"></textarea>
                </td>
            </tr>
            <tr>
                <td align="center" colspan="2">
                    <input type="button" value="提  交" onclick="check()">
                </td>
            </tr>
        </table>
</form>
```

（2）在该页的<script>标签中编写验证表单元素的值不允许为空的方法，关键代码如下：

```
function check(){
    var myform = document.getElementById("myform");  //获得 form 表单对象
    for(var i=0;i<myform.length;i++){                //循环 form 表单
        if(myform.elements[i].value==""){            //判断每一个元素是否为空
            alert(myform.elements[i].title+"不能为空！ ");
            myform.elements[i].focus();              //元素获得焦点
            return ;
        }
    }
    myform.submit();
}
```

秘笈心法

在 JavaScript 中，form 表单对象的 elements 属性的 value 属性表示指定元素的值，name 属性表示指定表单元素的名称，title 属性表示表单元素的标题。

实例 209　验证是否为数字　初级

光盘位置：光盘\MR\09\209　实用指数：★★★★

实例说明

在网站中填写注册信息时，有时可能会将数字类型的数据填写错误。本实例将介绍如何验证表单中某个元素的值是否为数字，实例运行效果如图 9.4 所示。

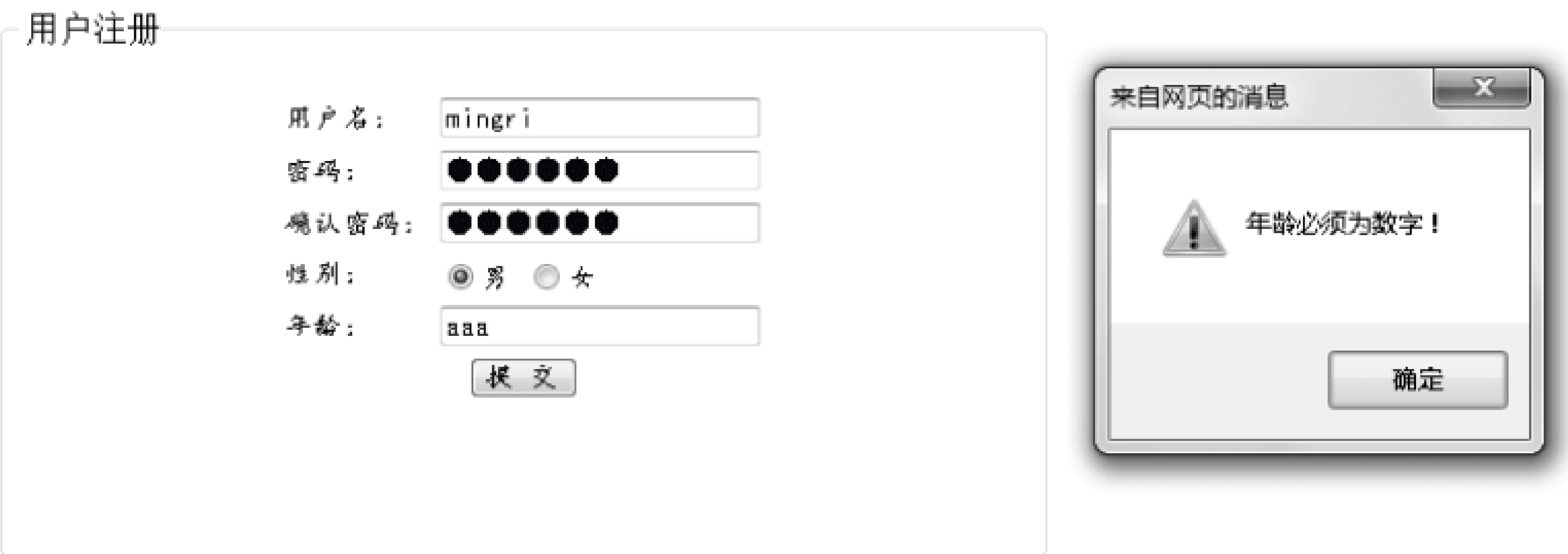


图 9.4　验证是否为数字

关键技术

验证是否为数字主要使用的是 JavaScript 的内置函数 isNaN()。该函数接收一个字符串类型的参数，如果该参数不是数字，则 isNaN()方法返回 true，否则返回 false。

设计过程

（1）新建用户注册表单页 index.jsp，该页中包含一个用户注册信息的表单，关键代码如下：

```
<form action="" id="myform">
        <table align="center">
                <tr>
                        <td>用户名：</td>
                        <td><input type="text"   id="name"></td>
                </tr>
                <tr>
                        <td>密码：</td>
                        <td><input type="password"   id="pwd"></td>
                </tr>
                <tr>
                        <td>确认密码：</td>
                        <td><input type="password"   id="pwd1"></td>
                </tr>
                <tr>
                        <td> 性别：</td>
                        <td>
                                <input type="radio" name="sex" id="man" value="m" />男
                                <input type="radio" name="sex" id="woman" value="f" />女
                        </td>
                </tr>
                <tr>
                        <td>年龄：</td>
                        <td><input type="text" id="age"></td>
                </tr>
                <tr>
                        <td align="center" colspan="2">
                                <input type="button" value="提 交" onclick="check()">
                        </td>
                </tr>
        </table>
    </form>
```

（2）在 index.jsp 页的<script>标签中编写表单提交按钮的 onClick 事件所调用的 JavaScript 方法。在 JavaScript 方法中获得表单中的年龄文本框的值，然后判断年龄是否为数字，如果是数字则提交表单，否则弹出提示信息，关键代码如下：

```
function check(){
        var age = document.getElementById("age");
        if(age.value==null||age.value==""){
                alert("请输入年龄！ ");
                age.focus();
                return;
        }
        if(isNaN(age.value)){
                alert("年龄必须为数字！ ");
                age.focus();
                return;
        }
        document.getElementById("myform").submit();
}
```

秘笈心法

在 JavaScript 中，常常需要通过 isNaN()函数来验证表单元素的值是否为数字，因此读者应该掌握该函数的含义和用法。

实例 210　验证 E-mail 是否正确

光盘位置：光盘\MR\09\210

初级

实用指数：★★★★

实例说明

在一些网站的用户注册的表单或其他表单中，经常会包含一个输入 E-mail 地址的文本框。如果用户将注册密码丢失或忘记时，需要提交密保问题来找回密码，如果密保问题正确，网站服务器会发一份带有用户密码的 E-mail 给用户，因此一个正确的 E-mail 地址非常重要。本实例将介绍如何通过 JavaScript 的正则表达式来验证 E-mail 地址是否正确。运行本实例，如图 9.5 所示，输入用户注册信息，在最后的 E-mail 地址中输入一个错误的地址，会弹出 E-mail 地址错误的提示信息。



图 9.5　验证 E-mail 是否正确

关键技术

本实例主要是应用 JavaScript 的正则表达式来实现的，验证 E-mail 地址的正则表达式如下：

```
var regExpression = /\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*/;
```

正则表达式的结构以正斜线“/”开头和结尾，一个正则表达式就是由普通字符（如字符 a～z）以及特殊字符（称为元字符）组成的文字模式。该模式描述在查找文字主体时待匹配的一个或多个字符串。正则表达式作为一个模板，将某个字符模式与所搜索的字符串进行匹配。常用的元字符及其在正则表达式上下文中的行为的说明如表 9.1 所示。

表 9.1　常用的元字符及其在正则表达式上下文中的行为的说明

字　符	说　明
\	将下一个字符标记为一个特殊字符，或一个原义字符，或一个后向引用，或一个八进制转义符。例如，'n' 匹配字符 "n"。'\n' 匹配一个换行符。序列 '\\' 匹配 "\" ，而 "\(" 则匹配 "("
^	匹配输入字符串的开始位置。如果设置了 RegExp 对象的 Multiline 属性，^ 也匹配 '\n' 或 '\r' 之后的位置
$	匹配输入字符串的结束位置。如果设置了 RegExp 对象的 Multiline 属性，$ 也匹配 '\n' 或 '\r' 之前的位置
*	匹配前面的子表达式零次或多次。例如，zo* 能匹配 "z" 以及 "zoo"。 * 等价于{0,}
+	匹配前面的子表达式一次或多次。例如，'zo+' 能匹配 "zo" 以及 "zoo"，但不能匹配 "z"。+ 等价于 {1,}
?	匹配前面的子表达式零次或一次。例如，"do(es)?" 可以匹配 "do" 或 "does" 中的"do" 。? 等价于 {0,1}
{*n*}	*n* 是一个非负整数。匹配确定的 *n* 次。例如，'o{2}' 不能匹配 "Bob" 中的 'o'，但是能匹配"food"中的两个
{*n*,}	*n* 是一个非负整数。至少匹配 *n* 次。例如，'o{2,}' 不能匹配 "Bob" 中的 'o'，但能匹配"foooood"中的所有 o。'o{1,}' 等价于 'o+'。'o{0,}' 则等价于 'o*'
{*n*,*m*}	*m* 和 *n* 均为非负整数，其中 *n* <= *m*。最少匹配 *n* 次且最多匹配 *m* 次。"o{1,3}"将匹配 "fooooood" 中的前 3 个 o。'o{0,1}' 等价于 'o?'，请注意在逗号和两个数之间不能有空格

续表

字　　符	说　　明
.	匹配除"\n"之外的任何单个字符。要匹配包括'\n'在内的任何字符，请使用像'[.\n]' 的模式
\b	匹配一个单词边界，也就是指单词和空格间的位置。例如，'er\b'可以匹配"never"中的'er'，但不能匹配"verb"中的'er'
\B	匹配非单词边界。'er\B'能匹配"verb"中的'er'，但不能匹配"never"中的'er'
\d	匹配一个数字字符。等价于[0-9]
\D	匹配一个非数字字符。等价于[^0-9]
\f	匹配一个换页符。等价于\x0c 和\cL
\n	匹配一个换行符。等价于\x0a 和\cJ
\r	匹配一个回车符。等价于\x0d 和\cM
\s	匹配任何空白字符，包括空格、制表符、换页符等。等价于[\f\n\r\t\v]
\S	匹配任何非空白字符。等价于[^ \f\n\r\t\v]
\t	匹配一个制表符。等价于\x09 和\cI
\v	匹配一个垂直制表符。等价于\x0b 和\cK
\w	匹配包括下划线的任何单词字符。等价于'[A-Za-z0-9_]'
\W	匹配任何非单词字符。等价于'[^A-Za-z0-9_]'
\xn	匹配 n，其中 n 为十六进制转义值。十六进制转义值必须为确定的两个数字长。例如，'\x41'匹配"A"。'\x041'则等价于'\x04' & "1"。正则表达式中可以使用 ASCII 编码
\n	标识一个八进制转义值或一个后向引用。如果\n 之前至少有 n 个获取的子表达式，则 n 为后向引用；否则，如果 n 为八进制数字（0～7），则 n 为一个八进制转义值

设计过程

新建用户注册表单页 index.jsp，在该页中添加<script>标签，然后在<script>中编写验证 E-mail 地址是否正确的 JavaScript 函数，在该函数中使用了 JavaScript 内置函数 test()，该函数会测试字符串的模式是否与正则表达式的模式相匹配，如果匹配，test()函数返回 true，否则返回 false，关键代码如下：

```
function checkEmail(){
        var email = document.getElementById("email");
        if(email.value==null||email.value==""){                    //判断文本框是否为空
                alert("请输入 E-mail 地址！ ");
                email.focus();
                return;
        }
        var regExpression = /\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*/;
        if(!regExpression.test(email.value)){                    //通过 test()函数测试字符串是否与表达式的模式匹配
                alert("您输入的 E-mail 地址不正确！ ");
                email.focus();                                    //使文本框获得焦点
                return;
        }
        document.getElementById("myform").submit();
}
```

在验证字符串是否与正则表达式的模式匹配时，除了以上方法中使用的 test()函数，还可以使用 search()函数，search()函数指明是否存在相应的匹配。如果找到一个匹配，search()函数将返回一个整数值，指明这个匹配距离字符串开始的偏移位置；如果没有找到匹配，则返回-1，关键代码如下：

```
function checkEmail(){
        var email = document.getElementById("email");
        if(email.value==null||email.value==""){                    //判断文本框是否为空
                alert("请输入 E-mail 地址！ ");
                email.focus();
                return;
```

```
        }
        var regExpression = /\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*/;
        var flag = email.value.search(regExpression);
        if(flag==-1){
                alert("您输入的 E-mail 地址不正确！ ");
                email.focus();
                return;
        }
        document.getElementById("myform").submit();
}
```

秘笈心法

在实际应用中，经常需要通过正则表达式来验证表单数据，因此，笔者希望读者能够掌握常用的元字符在正则表达式中表示的含义。

实例 211	验证电话号码是否正确 光盘位置：光盘\MR\09\211	初级 实用指数：★★★☆

实例说明

本实例将介绍如何通过 JavaScript 的正则表达式来验证输入的电话号码是否正确。运行本实例，如图 9.6 所示，在“固定电话”文本框中输入一个错误的电话号码，会弹出提示信息。



图 9.6　验证电话号码是否正确

关键技术

验证电话号码的正则表达式如下：

```
var regExpression = /^(0\d{2,3}-)?(\d{7,8})(-(\d{3,}))?$/;
```

该表达式验证的主要是国内的固定电话号码。验证模式为：区号+座机电话号码+分机号。表达式中的“\d{2,3}”表示 2 位或 3 位的 0～9 的数字，所以前面加“0”表示 3 位或 4 位的区号（如 0431、010、0451），“\d{7,8}”表示 7 位或 8 位的数字。有关元字符在正则表达式中的表示行为请参考实例 210 中的表 9.1 的说明。

设计过程

（1）新建添加通讯信息的表单页 index.jsp，关键代码如下：

```
<form action="" id="myform">
        <table align="center">
                <tr>
                        <td>姓名：</td>
                        <td><input type="text"   id="name"></td>
                </tr>
                <tr>
                        <td> 性别：</td>
                        <td>
                                <input type="radio" name="sex" id="man" value="m" />男
```

```
                <input type="radio" name="sex" id="woman" value="f" />女
            </td>
        </tr>
        <tr>
            <td>年龄：</td><td><input type="text" id="age"></td>
        </tr>
        <tr>
            <td>电子邮箱：</td><td><input type="text" id="email"></td>
        </tr>
        <tr>
            <td>固定电话：</td><td><input type="text" id="telephone"></td>
        </tr>
        <tr>
            <td>联系地址：</td><td><textarea rows="5" cols="30"></textarea></td>
        </tr>
        <tr>
            <td align="center" colspan="2">
                <input type="button" value="提  交" onclick="checkPhone()">
            </td>
        </tr>
    </table>
</form>
```

（2）在该页的<script>标签中编写验证电话号码的 JavaScript 函数，如果验证成功则提交表单，否则弹出提示信息，关键代码如下：

```
function checkPhone(){
    var telephone = document.getElementById("telephone");
    var regExpression = /^(0\d{2,3}-)?(\d{7,8})(-(\d{3,}))?$/;
    if(!regExpression.test(telephone.value)){
            alert("您输入的固定电话有误！");
        telephone.focus();
        return;
    }
    document.getElementById("myform").submit();
}
```

秘笈心法

根据本实例的实现，读者可以实现在企业客户管理系统中添加客户信息时验证输入的联系电话是否合法，以及在通讯录管理系统中添加讯友信息时验证输入的电话号码。

实例 212	验证手机号码是否正确 光盘位置：光盘\MR\09\212	初级 实用指数：★★★★☆

实例说明

在网站中注册账号或其他操作时，可能需要填写个人的手机号码，为了数据的有效性，需要对填写的手机号码进行验证。本实例将介绍如何通过正则表达式来验证手机号码是否正确，运行结果如图 9.7 所示。



图 9.7　验证手机号码是否正确

关键技术

验证手机号码的 JavaScript 正则表达式如下：

```
/^(86)?((13\d{9})|(15[0,1,2,3,5,6,7,8,9]\d{8})|(18[0,5,6,7,8,9]\d{8}))$/;
```

该正则表达式是根据目前的手机号段定义的，目前的手机号段包括：130～139、150～159（除了 154 之外）、180、185～189。

设计过程

新建添加通讯信息的表单页 index.jsp，在该页中编写验证手机号码是否正确的正则表达式，关键代码如下：

```
function checkPhone(){
	var mobileNo = document.getElementById("mobileNo");
	var exp = /^(86)?((13\d{9})|(15[0,1,2,3,5,6,7,8,9]\d{8})|(18[0,5,6,7,8,9]\d{8}))$/;
	if(!exp.test(mobileNo.value)){
		alert("您输入的手机号码有误！");
		mobileNo.focus();
		return;
	}
	document.getElementById("myform").submit();
}
```

秘笈心法

在验证字符串是否符合指定的正则表达式模式时，除了使用 test()方法，还可以使用 search()方法判断字符串是否匹配正则表达式模式。如果找到一个字符串匹配正则表达式，search()方法将返回一个整数值，指明这个匹配距离字符串开始的偏移位置；如果没有找到匹配，search()方法返回-1。search()方法的语法结构如下：

```
stringObj.search(rgExp)
```

参数说明

❶ stringObj：必选参数。要被判断的字符串文字或 String 对象。

❷ rgExp：必选参数。包含正则表达式模式和可用标志的正则表达式对象。

实例 213 验证字符串是否为汉字

光盘位置：光盘\MR\09\213

初级

实用指数：★★★☆

实例说明

在 Web 应用开发过程中，有些表单元素中限制用户输入的数据必须为汉字，如用户的真实姓名。本实例将介绍如何验证用户输入的字符串是否为汉字，运行结果如图 9.8 所示。



图 9.8　验证字符串是否为汉字

关键技术

验证字符串是否为汉字的正则表达式如下：

```
var regExpression = /[\u4E00-\u9FA5]/;
```

在该表达式中“4E00”和“9FA5”代表汉字的 Unicode 编码，常用汉字的 Unicode 编码范围在“4E00”和“9FA5”之间。

设计过程

新建用户注册页 index.jsp，在该页的<script>标签中编写验证字符串是否为汉字的 JavaScript 函数，关键代码如下：

```
function checkRealName(){
	var realName = document.getElementById("realName");
	var regExpression = /[\u4E00-\u9FA5]/;
	if(!regExpression.test(realName.value)){
		alert("您输入的真实姓名不正确，必须为汉字！");
		realName.focus();
		return;
	}
	document.getElementById("myform").submit();
}
```

秘笈心法

根据本实例，读者可以实现限制在线订单中的联系人姓名必须为汉字，限制进销存管理系统中的经手人、联系人字段必须为汉字。

实例 214	验证身份证号码是否有效 光盘位置：光盘\MR\09\214	初级 实用指数：★★★☆

实例说明

在一些网站的会员注册或其他需要填写身份证号码的表单中，需要对身份证号码进行有效性验证。本实例将介绍如何通过正则表达式来验证身份证号是否有效，运行结果如图 9.9 所示。



图 9.9　验证身份证号码是否有效

关键技术

目前有效的身份证号有两种编码规则，分别是 15 位的编码和 18 位的编码，15 位的身份证号码的规则是 1985 年制订的，在 1999 年国家对新发行的身份证号码规则进行了修改，由 15 位改为 18 位，主要是把号码中的出生日期的两位年份（yy）改为 4 位年份（yyyy）。下面分别介绍这两种身份证号码的编码规则。

（1）15 位的身份证号。如 220122850912213，需要遵循以下规则：

- ❑ 1～6 位为地区代码，1、2 位数字为各省级政府的代码，3、4 位数字为地市级政府的代码，5、6 位数字为县、区级政府的代码。
- ❑ 7～12 位为身份证持有人的出生年月日（yymmdd）。
- ❑ 13～15 位为顺序号，并能够判断性别。

（2）18 位的身份证号码。如 220122198410222218，需要遵循以下规则：

- 1～6 位为地区代码，与 15 位的身份证号码前 6 位含义相同。
- 7～14 位为出生年月日（yyyymmdd）。
- 15～17 位为顺序码，为县、区级政府所辖派出所的分配码。
- 第 18 位为校验码，通过复杂的公式算出。校验码是 1～10 之间的数字，由于不能用 10 作为校验码，所以用 X 代替。

设计过程

（1）新建 index.jsp 页，在该页中编写验证身份证号码是否有效的 JavaScript 函数，关键代码如下：

```
function checkIDCard(){
	var IDCard = document.getElementById("IDCard");
	//验证 15 位身份证号码
	var regIDCard_15 = /^[1-9]\d{7}((0\d)|(1[0-2]))(([0|1|2]\d)|3[0-1])\d{3}$/;
	//验证 18 位身份证号码
	var regIDCard_18 = /^[1-9]\d{5}[1-9]\d{3}((0\d)|(1[0-2]))(([0|1|2]\d)|3[0-1])\d{3}[\d|x|X]$/;
	if(IDCard.value.length!=15&&IDCard.value.length!=18){
		alert("您输入的身份证号码长度不对，请输入 15 位或 18 位的身份证号码！");
		IDCard.focus();
		return;
	}else{
		if(IDCard.value.length==15){	//验证 15 位的身份证号码
			if(!regIDCard_15.test(IDCard.value)){
				alert("您输入的身份证号码有误！");
				IDCard.focus();
				return;
			}
		}
		if(IDCard.value.length==18){	//验证 18 位的身份证号码
			if(!regIDCard_18.test(IDCard.value)){
				alert("您输入的身份证号码有误！");
				IDCard.focus();
				return;
			}
		}
	}
	document.getElementById("myform").submit();
}
```

（2）在 web.xml 中配置 CounterServlet 类，关键代码如下：

```
<servlet>
	<servlet-name>CounterServlet</servlet-name>
	<servlet-class>com.lh.servlet.CounterServlet</servlet-class>
</servlet>
<servlet-mapping>
	<servlet-name>CounterServlet</servlet-name>
	<url-pattern>/counter</url-pattern>
</servlet-mapping>
```

秘笈心法

本实例在验证身份证号码时，只是验证其基本逻辑规则，并没有具体验证身份证号码中的前 6 位的地区代码是否正确，以及表示出生日期的号码是否符合日期规则。读者可以在本实例实现的基础上，实现验证身份证号码的前 6 位是否符合地区代码的规则以及出生日期的规则。具体的地区代码可以查找相关资料。

实例 215	验证车牌号码是否有效 光盘位置：光盘\MR\09\215	初级 实用指数：★★★☆

实例说明

在 Web 应用的车辆管理的功能模块中，在添加车辆信息时，需要对车牌号码进行有效性验证。本实例将介

绍如何通过 JavaScript 正则表达式来验证车牌号码是否有效，程序运行结果如图 9.10 所示。



图 9.10　验证车牌号码是否有效

关键技术

本实例实现验证的是国内的非军用车的车牌号码，验证规则如下：

（1）第一个字符必须为汉字。这个汉字代表全国各省、自治区、直辖市名称的简写。

（2）第二个字符必须为英文字母。代表各省、自治区、直辖市内的不同的市级编号，如吉林省的长春市字母为 A、吉林市为 B、松原市为 J 等。

（3）第 3 个字符为分隔符“-”。

（4）4～8 位为数字和字母的组合。第 4 位和第 5 位可能为数字或字母，例如，吉 A-E5678、吉 A-ER123、吉 A-6A789、吉 A-66A78、吉 A-888888 等。第 8 位可能为字母，如吉 A-1001A。

设计过程

新建添加车辆信息的表单页 index.jsp，在该页的<script>标签中编写验证车牌号码是否有效的 JavaScript 函数，该函数主要验证非军用车牌号码，并且验证车牌号的首字符的汉字是否为各省名称的简写，关键代码如下：

```
function checkNumberPlate(){
	var numberPlate = document.getElementById("numberPlate");
	var shortProvince = new Array ("京","沪","津","渝","冀","晋","蒙",
	"辽","吉","黑","苏","浙","皖","闽","赣","鲁","豫","鄂","湘","粤","桂",
	"琼","川","贵","云","藏","陕","甘","青","宁","新");
	if(numberPlate.value==""||numberPlate.value==null){  //验证是否为空
		alert("请输入车牌号码！");
		numberPlate.focus();
		return;
	}
	var str = numberPlate.value.substring(0,1);              //截取车牌号的首字符
	var res = false;
	for(var i=0;i<shortProvince.length;i++){                //循环判断输入的车牌号首字符是否在指定的省份内
		if(str==shortProvince[i].toString()){
			res = true;                                     //在指定简写省份名称范围内
		}
	}
	//验证普通的车牌号码，如吉 A-E1234、吉 A-EE123、吉 A-12345、吉 A-6A123
	var regExpression1 = /^[\u4e00-\u9fa5][a-zA-Z]-[0-9A-Za-z]{3}\d{2}$/;
	//验证车牌号尾号为字母的表达式
	var regExpression2 = /^[\u4e00-\u9fa5][a-zA-Z]-\d{4}[a-zA-Z]$/;
		//判断用户输入的车牌号是否符合指定表达式的规则
	if(!regExpression1.test(numberPlate.value)&&
		!regExpression2.test(numberPlate.value)){
		alert("您输入的车牌号码有误！");
		numberPlate.focus();
		return;
	}else{//如果符合规则，判断首字符是否为各省级名称的简写
		if(!res){
			alert("您输入的车牌号码有误，首字符无效！");
			numberPlate.focus();
			return;
		}
```

```
        }
        document.getElementById("myform").submit();
}
```

秘笈心法

由于车辆不断增多，车牌号码规则可能也会随之改变，本实例实现的只是车牌号码的暂时规则，而并非最终的车牌号码规则，读者可以根据具体情况来修改车牌号码的正则表达式规则。

实例 216　验证网站地址是否有效

光盘位置：光盘\MR\09\216

初级　实用指数：★★★☆

实例说明

在有些网站的提交表单中可能会包含用户输入网站地址的文本框，为了保证数据的有效性，需要对网站地址进行合法验证。本实例将介绍如何验证用户输入的网站地址是否有效，运行效果如图 9.11 所示。



图 9.11　验证网站地址是否有效

关键技术

验证网站域名地址是否合法是通过 JavaScript 的正则表达式实现的。一般的网站 URL 地址可能是一个 IP 地址（如 192.168.10.16）、域名（如 www.sina.com）、域名的简写形式（如 www.baidu.com，可以写成 baidu）等。有关元字符在正则表达式中的表示行为请参考实例 210 中的表 9.1 的说明，此处不再赘述。

设计过程

新建 index.jsp 表单页，在表单中包含一个输入网站 URL 地址的文本框，然后表单的“提交”按钮的 onClick 事件会调用验证输入的网站 URL 是否有效的 JavaScript 函数，该函数的关键代码如下：

```
function checkNetURL(){
        var netURL = document.getElementById("netURL");
        var regStr = "^((https|http|ftp|rtsp|mms)?://)"
                + "(([0-9a-zA-Z_!~*'().&=+$%-]+: )?[0-9a-zA-Z_!~*'().&=+$%-]+@)?" //ftp 的 user@
                + "(([0-9]{1,3}\.){3}[0-9]{1,3}"                    //验证 IP 形式的 URL，如 192.168.10.16
                + "|"                                               //输入的可以是 IP 或域名
                + "([0-9a-zA-Z_!~*'()-]+\.)*"                       //验证域名 www.
                + "([0-9a-zA-Z][0-9a-zA-Z-]{0,61})?[0-9a-zA-Z]\."   //验证二级域名
        + "[a-zA-Z]{2,6})(:[0-9]{1,4})?"                            //域名中可能包含端口
                + "((/?)|"
                + "(/[0-9a-zA-Z_!~*'().;?:@&=+$,%#-]+)+/?)$";
        var re=new RegExp(regStr);                                  //创建正则表达式对象
        if(!re.test(netURL.value)){                                 //验证输入的字符串是否符合正则表达式规则
                alert("您输入的网站 URL 地址有误！ ");
                netURL.focus();
                return;
        }
        document.getElementById("myform").submit();
}
```

秘笈心法

由于本实例中编写的正则表达式过长，所以在编写正则表达式时，并没有在正则表达式的开始位置以及结尾使用斜杠“/”，而是使用“+”连接符生成一个长字符串，这个长字符串需要通过 RegExp 对象转换为正则表达式。

实例 217	验证数量和金额 光盘位置：光盘\MR\09\217	初级 实用指数：★★★☆

实例说明

在 Web 应用开发或者网站开发时，经常会遇到限制用户输入数据的类型的问题，例如，表示“数量”的文本框只能输入正整数，“金额”文本框只能输入数值型数据。本实例将介绍如何验证用户输入的数量和金额是否正确，运行结果如图 9.12 所示，当用户输入不合法的金额或数量时，将弹出提示对话框，提示用户输入错误。

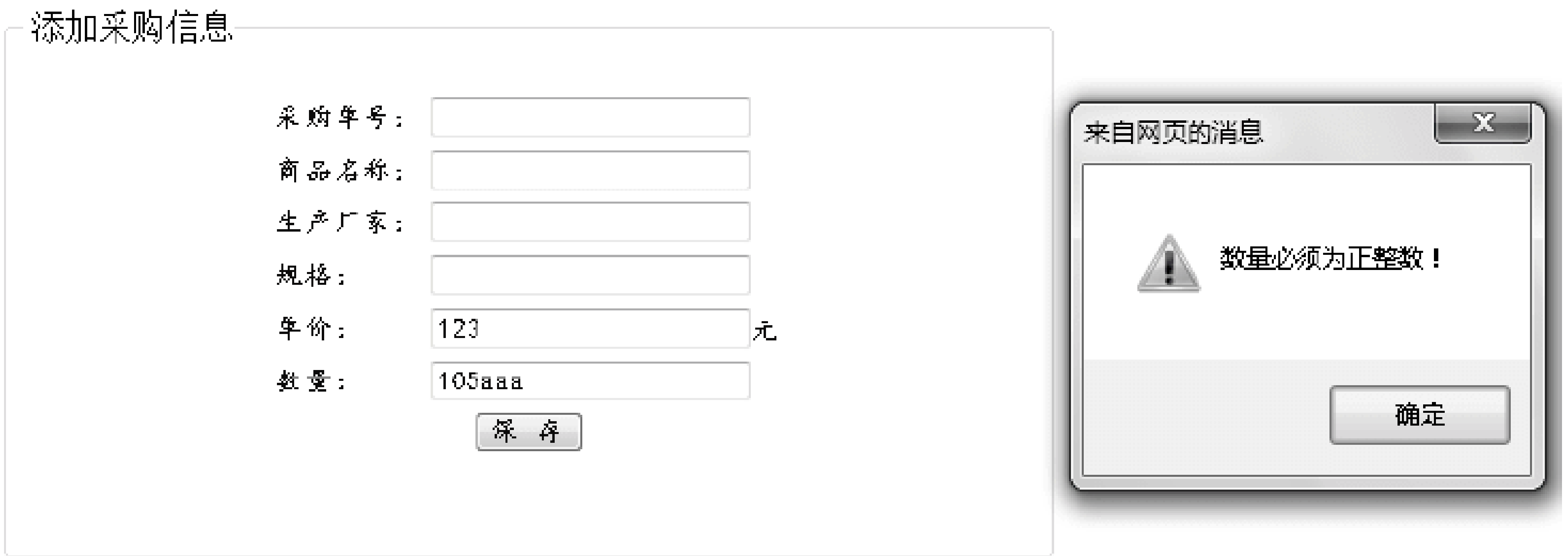


图 9.12　验证数量和金额

关键技术

验证文本框中输入的数量时，应该限制用户只能输入正整数，可以通过正则表达式来验证。验证数量的正则表达式如下：

```
var regExp = /^[1-9]+(\d*)$/;                        //验证数量的正则表达式
```

验证金额时，应该限制用户输入的金额为数值型数据，可以包含小数部分，可以通过 JavaScript 中的 isNaN() 方法来验证，该方法返回一个 Boolean 值，指明提供的值是否是保留值 NaN（不是数字）。如果值是 NaN，那么 isNaN 函数返回 true，否则返回 false。

设计过程

（1）新建 index.jsp 页，在该页的<script>标签中编写验证数量和金额的 JavaScript 函数，关键代码如下：

```
function checkInsert(){
        var number = document.getElementById("number");    //数量文本框对象
        var price = document.getElementById("price");      //单价文本框对象
        var regExp2 = /^[1-9]+(\d*)$/;                     //验证数量的正则表达式
        if(isNaN(price.value)){                            //验证单价是否为数字
                alert("您输入的单价不是有效值！");
                price.focus();
                return false;
        }
        if(number.value!=null&&number.value!=""){
                if(!regExp2.test(number.value)){           //验证数量是否符合正则表达式
                        alert("数量必须为正整数！");
                        number.focus();
                        return false;
                }
        }
```

```
        return true;
}
```

（2）编写表单提交按钮的 onClick 事件所调用的方法，关键代码如下：

```
function save(){
        if(checkInsert()){
                document.getElementById("myform").submit();
        }
}
```

秘笈心法

根据本实例，读者可以实现在商品订单表单中验证商品的数量和商品的价格。

实例 218　验证字符串是否以指定字符开头

光盘位置：光盘\MR\09\218

初级

实用指数：

实例说明

本实例将介绍如何通过 JavaScript 正则表达式，来验证一个字符串是否以指定字符开头，运行结果如图 9.13 所示。



图 9.13　验证字符串是否以指定字符开头

关键技术

验证字符串是否以指定字符开头，主要通过 JavaScript 正则表达式来实现，与前面讲解的实例中所使用的正则表达式不同，本实例中使用的正则表达式是动态改变的，写法如下：

```
var startStr = document.getElementById("startStr").value;
var regExp=eval("/^"+startStr+"/");
```

在正则表达式中使用变量时，应该通过 JavaScript 提供的内置函数 eval()来动态执行。eval()函数允许 JavaScript 源代码的动态执行，也就是说首先会获得变量的值，然后再将整个字符串组合成一个正则表达式。

设计过程

（1）新建 index.jsp 页，在该页的<script>标签中编写验证字符串是否以指定字符开头的 JavaScript 函数，关键代码如下：

```
function checkStr(){
        var startStr = document.getElementById("startStr").value;
        var str = document.getElementById("str").value;
        var regExp=eval("/^"+startStr+"/");              //使用 eval()方法使 JavaScript 动态执行
                if(startStr==""){                         //验证输入的字符串是否为空
                alert("请输入字符串的起始字符！");
                document.getElementById("startStr").focus();
                return false;
        }
        if(str==""){                                      //验证输入的字符串是否为空
                alert("请输入要判断的字符串！");
                document.getElementById("str").focus();
                return false;
        }
        if(!regExp.test(str)){                            //字符串不是以指定字符开头
```

```
            alert("指定的字符串不是以 ["+startStr+"] 开头！");
            return false;
        }else{                                      //字符串是以指定字符开头
            alert("指定的字符串是以 ["+startStr+"] 开头！");
            return true;
        }
        return true;
}
```

（2）编写表单提交按钮的 onClick 事件所调用的方法，在该方法中调用验证字符串是否以指定字符开头的函数 checkStr()，如果该函数返回 true，则提交表单，否则会弹出错误提示信息，关键代码如下：

```
function check(){
        if(checkStr()){
            document.getElementById("myform").submit();
        }
}
```

秘笈心法

本实例实现的关键是在 JavaScript 中使用 eval()方法，该方法允许 JavaScript 源代码的动态执行。

实例 219　限制输入字符串的长度

光盘位置：光盘\MR\09\219

高级

实用指数：★★★★

实例说明

在一些表单提交之前，为了确保提交数据的有效性，有些输入文本框的信息是需要限制长度的。例如，用户注册表单中的用户名长度，文本域中输入的字符串长度等。本实例将介绍如何限制用户输入的字符串长度。运行本实例，如图 9.14 所示，在"用户名"文本框中输入超过 10 个字符的字符串，单击"注册"按钮时，将弹出提示信息。

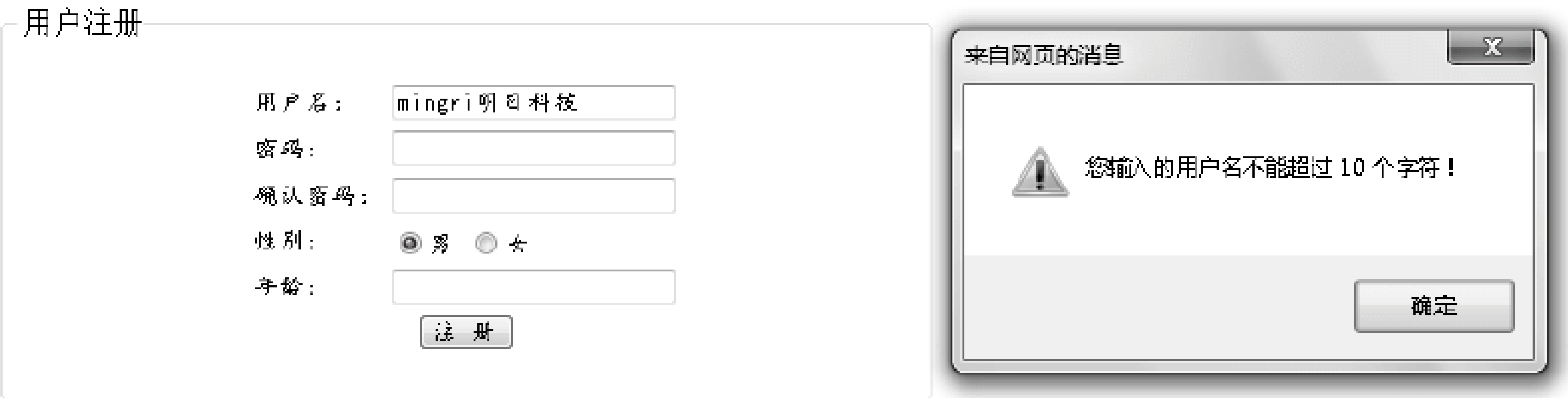


图 9.14　限制输入字符串的长度

关键技术

用户在文本框中输入字符串时，可能会输入包含汉字的字符串，因为汉字是占两个字节的，所以在计算字符串的长度时，使用 JavaScript 的 String 对象的 length 属性来获取字符串的长度是不准确的。应该使用 String 对象的 charCodeAt()函数，该函数可以将字符串中的指定字符转换为 Unicode 编码，通过字符的 Unicode 编码范围来判断字符串中的汉字的长度。

设计过程

（1）新建用户注册表单页 index.jsp，在该页的<script>标签中编写限制输入字符串长度的 JavaScript 函数，关键代码如下：

```
/**
*限制输入字符串的长度
*@ str：要判断的字符串
*@ limitLength：限制的长度
*/
```

```
function checkStrLength(str,limitLength){
	var n=0;                          //该变量保存字符串的长度
	for(var i=0;i<str.length;i++){
		var code = str.charCodeAt(i);     //获得每个字符的 Unicode 值
		if(code>255){
			n=n+2;
		}
		else{
			n=n+1;
		}
	}
	if(n>limitLength){                //如果字符串的长度大于限制长度，返回 false
		return false;
	}
	return true;
}
```

（2）编写表单提交按钮的 onClick 事件所调用的函数，在该函数中验证输入用户名的长度，关键代码如下：

```
function check(){
	var userName = document.getElementById("username").value;
	var limitNum = 10;
	if(userName==""){
		alert("请输入用户名！");
		document.getElementById("username").focus();
		return;
	}
	if(!checkStrLength(userName,10)){
		alert("您输入的用户名不能超过 "+limitNum+" 个字符！");
		return;
	}
	document.getElementById("myform").submit();
}
```

秘笈心法

由于表单数据最终是要保存在数据库中的，而数据库中保存字符串类型数据的字段长度是根据字节计算的，例如，在 MySQL 数据库中某个信息表的字段类型为 varchar(10)，那么这个字段只能保存 5 个中文字符。因此，通过 JavaScript 计算字符串长度时，应该使用 charCodeAt()函数判断字符串中是否包含中文字符。

实例 220	验证输入字符串是否包含特殊字符 光盘位置：光盘\MR\09\220	初级 实用指数：★★★★

实例说明

在 Web 应用开发过程中，有些表单数据是不允许包含特殊字符的。本实例将介绍如何通过 JavaScript 来验证字符串是否包含特殊字符，运行结果如图 9.15 所示。



图 9.15　验证字符串是否包含特殊字符

关键技术

本实例主要通过 JavaScript 正则表达式来实现。验证特殊字符的正则表达式的写法如下：

```
var regExpress = /[\'\"\\<>;&=#]/;
```

设计过程

（1）新建 index.jsp 页，在该页中编写验证字符串中是否包含特殊字符的 JavaScript 函数，关键代码如下：

```
function checkEspecialCode(str){
        var regExpress = /[\'\"\\<>;&=#]/;          //验证特殊字符的正则表达式
        if(regExpress.test(str)){                    //测试字符串是否包含指定的特殊字符
                return false;
        }
        return true;
}
```

（2）编写表单提交按钮的 onClick 事件所调用的 JavaScript 函数，关键代码如下：

```
function check(){
        var userName = document.getElementById("username").value;
        if(userName==""){
                alert("请输入用户名！");
                document.getElementById("username").focus();
                return;
        }
        if(!checkEspecialCode(userName)){
                alert("您输入的用户名中包含无效字符！");
                document.getElementById("username").focus();
                return;
        }
        document.getElementById("myform").submit();
}
```

秘笈心法

限制输入字符串不能包含特殊字符的目的是防止 SQL 注入，因此通过 JavaScript 实现限制输入的字符串不允许包含特殊字符是非常有必要的。

实例 221　限制用户不允许输入中文字符

初级

光盘位置：光盘\MR\09\221

实用指数：★★★★

实例说明

在网站开发过程中，有些表单中的信息经常会限制不允许包含中文字符。本实例将介绍如何通过 JavaScript 来限制输入字符串中不允许包含中文字符，运行效果如图 9.16 所示。

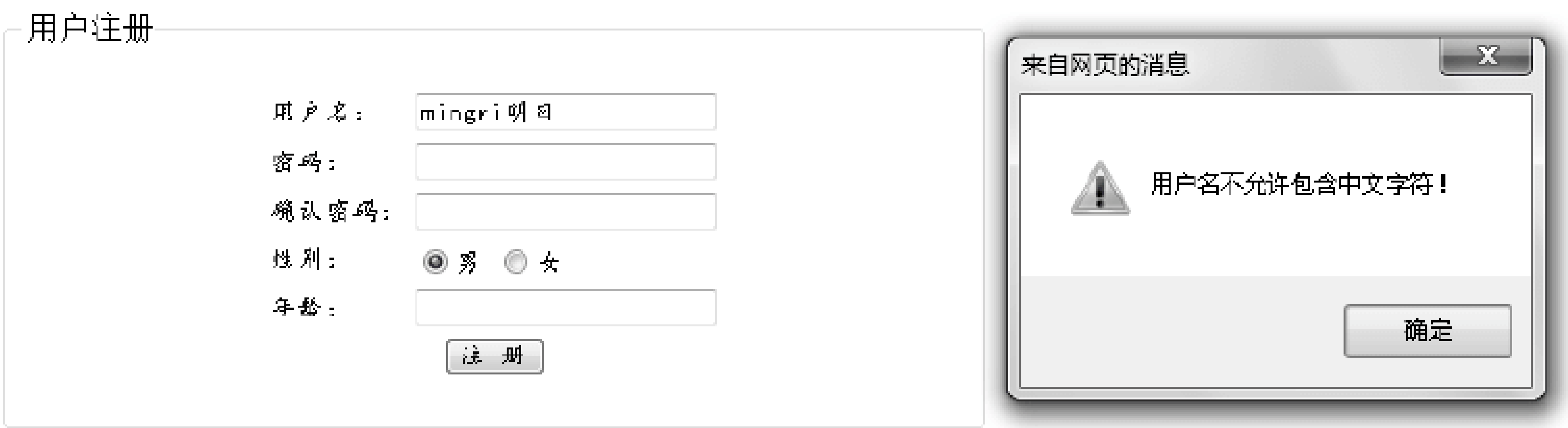


图 9.16　限制用户不允许输入中文字符

关键技术

本实例的实现主要应用 JavaScript 提供的 excape()方法。该方法返回一个包含了 charstring 内容的字符串值（Unicode 格式）。所有空格、标点、重音符号以及其他非 ASCII 字符都用%*xx* 编码代替，其中 *xx* 表示该字符的十六进制数。例如，空格返回的是“%20”。字符值大于 255 的以%u*xxxx* 格式存储，而大于 255 的表示中文字符，所以，通过判断转换后的字符串是否包含“%u”就能够知道字符串中是否包含中文字符。

设计过程

（1）新建 index .jsp 页，在该页中编写判断字符串中是否包含中文字符的 JavaScript 函数，关键代码如下：

```
function checkCN(str){
	if (escape(str).indexOf("%u")<0){//indexOf()方法查找字符串是否包含"\u"，如果不包含返回-1
		return true;
	} else {
		return false;
	}
}
```

（2）编写表单提交按钮 onClick 事件调用的 JavaScript 函数，关键代码如下：

```
function check(){
	var userName = document.getElementById("username").value;
	if(userName==""){
		alert("请输入用户名！");
		document.getElementById("username").focus();
		return;
	}
	if(!checkCN(userName)){
		alert("用户名不允许包含中文字符！");
		document.getElementById("username").focus();
		return;
	}
	document.getElementById("myform").submit();
}
```

秘笈心法

根据本实例，读者可以实现限制网站用户登录时，用户名不允许包含中文。

9.2 字符串处理

字符串在程序中的应用非常多，是程序中不可缺少的部分。本节将介绍一些在程序中常用于字符串处理的相关实例。

实例 222 小写金额转换为大写金额

光盘位置：光盘\MR\09\222

初级

实用指数：★★★★

实例说明

在开发财务或与金融相关的网络管理系统时，经常会涉及金额的大小写转换问题。在进行账目输入时，常采用输入小写金额的方式方便用户，但是它可能存在被篡改的问题，这对于金融系统来说是不安全的，解决办法是将用户输入的小写金额转换为大写。运行本实例，如图 9.17 所示，输入小写金额，单击“转换”按钮后会转换为大写。

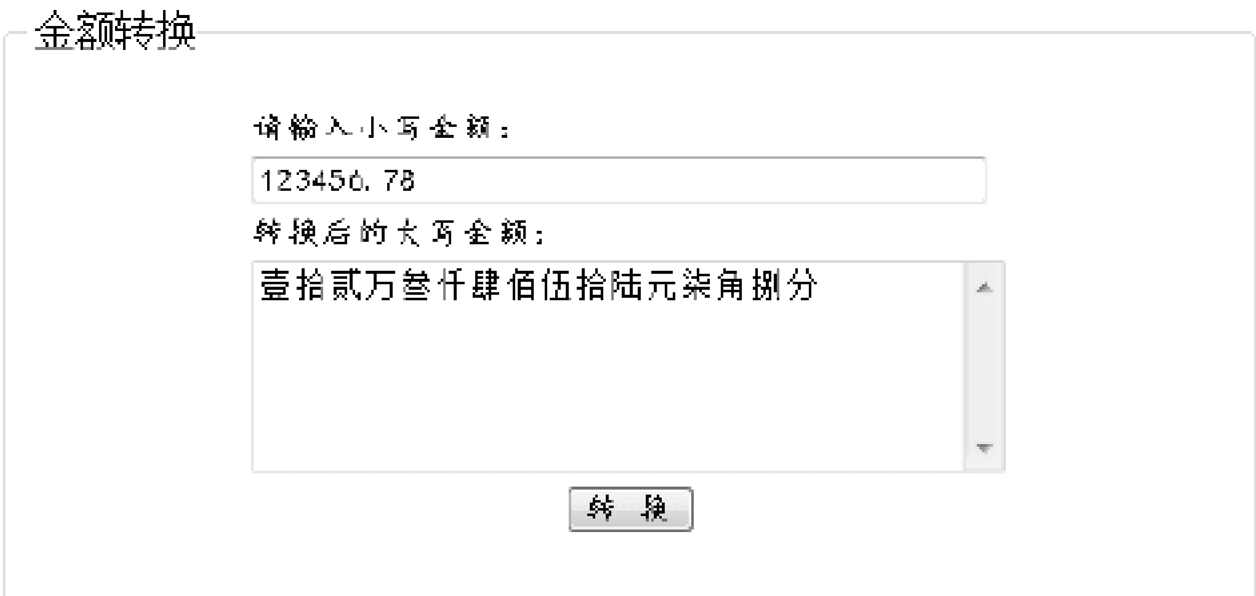


图 9.17　小写金额转换为大写金额

关键技术

本实例通过使用 JavaScript 的实现思路如下：

（1）将大写的数字和单位分别保存在两个字符串中。

（2）去除用户输入的小写金额中的小数点。

（3）根据用户输入的小写金额长度取出该字符串所用到的单位。

（4）取出小写字符对应的大写字符，并与单位组合成新的字符串，同时还要对小写金额中存在连续多个 0 的情况进行处理。

设计过程

（1）首先对用户输入的小写金额的合法性进行判断，然后对合法数据的小数位数进行控制，再调用自定义的 JavaScript 函数将小写金额转换为大写金额，关键代码如下：

```
function convert(){
	var money_num = document.getElementById("money_num").value;
	if(money_num==""){
		alert("请输入金额！");
		document.getElementById("money_num").focus();
		return;
	}
	if(isNaN(money_num)){
		alert("请输入数字类型的金额 !");
		return;
	}
	if(money_num>999999999999){
		alert("您输入的金额不能大于 999999999999!");
		return;
	}
	//将小数点后保留两位小数
	if(money_num.indexOf(".")>0){
		var decimalStr = money_num.split(".");
		if(decimalStr[1].length>2){
			decimalStr[1]=decimalStr[1].substr(0,2);
		}
		money_num = decimalStr[0]+"."+decimalStr[1];
	}
	value=change(money_num);                                //调用自定义函数转换
	document.getElementById("money_cn").value=value;        //将转换后的值赋给文本框
}
```

（2）将小写金额转换为大写金额的自定义 JavaScript 函数的代码如下：

```
function change(str){
	je="零壹贰叁肆伍陆柒捌玖";                              //大写的数字（0～9）
	cdw="万仟佰拾亿仟佰拾万仟佰拾元角分";                    //金额单位
	var newstring=(str*100).toString();                     //将金额值乘以 100
	newstringlog=newstring.length;                          //乘以 100 之后的金额的长度
	newdw=cdw.substr(cdw.length-newstringlog);
	num0=0;                                                 //记录零的个数
	wan=0;                                                  //记录万位出现的次数
	dxje="";                                                //记录大写金额
	for(m=1;m<newstringlog+1;m++){
		xzf=newstring.substr(m-1,1);
		dzf=je.substr(xzf,1);
		dw=newdw.substr(m-1,1);
		if(dzf=="零"){
			dzf="";
		if(dw=="亿"){
		}else if(dw=="万"){
			dzf="";
			wan=1;
		}else if(dw=="元"){

		}else{
```

```
                        dw="";          //记录单位
                }
                num0=num0+1;
                }else{
                        if(num0-wan>0){
                                if(dw!="角"){
                                dzf="零"+dzf;
                                }
                        }
                        num0=0;

                }
                dxje=dxje+dzf+dw;
        }
        if(newstring.length!=1){
                if(newstring.substr(newstring.length-2)=="00"){
                        dxje=dxje+"整";
                }else{
                        dxje=dxje;
                }
        }
        return dxje;
}
```

秘笈心法

根据本实例，读者可以实现销售单中销售金额的大小写转换、汇款金额的大小写转换以及物资采用单中金额的大小写转换。

实例 223 去掉字符串左右空格

光盘位置：光盘\MR\09\223

初级

实用指数：★★★★

实例说明

用户在填写提交表单时，可能因为误操作在文本框中输入的字符串的开头和结尾处输入了空格，而服务器端又不想获得这个带有左右空格的表单信息，因此在表单提交之间应该对用户输入的不必要的空格进行处理。运行本实例，如图 9.18 所示，输入一个带有左右空格的字符串，单击“去掉左右空格”按钮后，会去掉字符串的左右空格并将其显示在“转换后的字符串”文本框中。

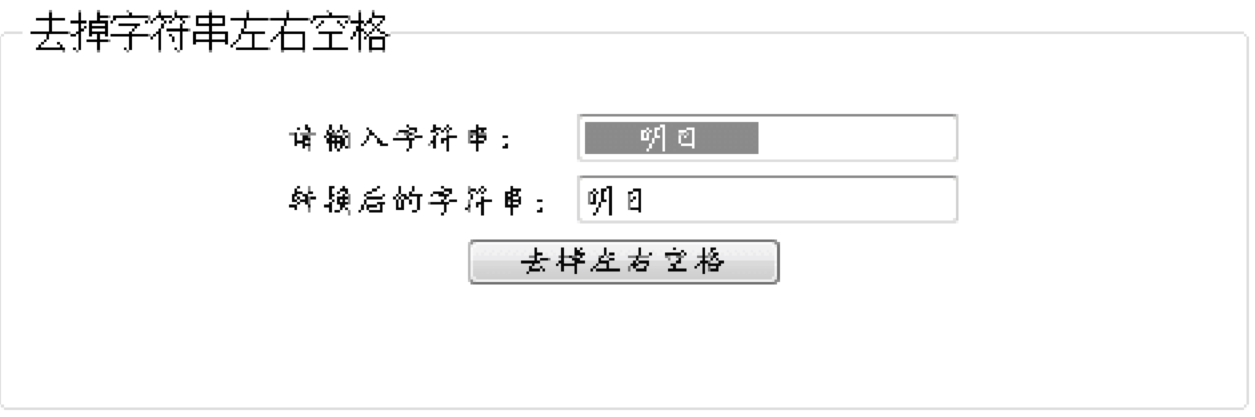


图 9.18　去掉字符串左右空格

关键技术

本实例主要是通过 JavaScript 正则表达式和 JavaScript 中的 replace()方法实现的。在正则表达式中使用“\s”来匹配任何空白字符，包括空格、制表符、换页符等。验证字符串中是否包含左右空格的正则表达式如下：

```
var regExp = /(^\s*)|(\s*$)/g;
```

replace()方法的结果是一个完成了指定替换的 stringObj 对象的复制，其语法结构如下：

```
stringObj.replace(rgExp, replaceText)
```

参数说明

❶ stringObj：要执行该替换的 String 对象或字符串文字。该字符串不会被 replace()方法修改。

❷ rgExp：为包含正则表达式模式或可用标志的正则表达式对象，也可以是 String 对象或文字。如果 rgExp

不是正则表达式对象，它将被转换为字符串，并进行精确的查找；不要尝试将字符串转化为正则表达式。

❸ replaceText：是一个 String 对象或字符串文字，对于 stringObj 中每个匹配 rgExp 的位置都用该对象所包含的文字加以替换。

设计过程

（1）新建 index.jsp 页，编写去掉字符串左右空格的 JavaScript 函数，该函数返回去掉空格后的字符串，关键代码如下：

```
function trim(str){
	var regExp = /(^\s*)|(\s*$)/g; //验证左右空格的正则表达式
	str = str.replace(regExp,"");   //去掉字符串的左右空格
	return str;
}
```

（2）编写表单按钮 onClick 事件所调用的 JavaScript 自定义函数，再将去掉空格后的字符串赋给文本框，关键代码如下：

```
function convert(){
	var str = document.getElementById("str").value;
	if(str==""){
		alert("请输入字符串！ ");
		document.getElementById("str").focus();
		return;
	}
	document.getElementById("convertStr").value = trim(str);
}
```

秘笈心法

如果保存在数据库中的某个字段的数据带有空格，当在程序中根据该字段实现查询数据时，由于当时输入保存数据时不小心输入了空格，那么在根据该字段查询数据时，由于存在空格可能导致没有查询结果。因此，本实例的实现是很有必要的。

实例 224　将数字字符串格式化为指定长度

光盘位置：光盘\MR\09\224

初级　实用指数：★★★★

实例说明

在实际的开发过程中，经常需要将输入的数字格式化为指定的长度，如果不足指定位数时，在该数字的前面用“0”补齐。运行本实例，如图 9.19 所示，在文本框中输入要格式化的数字和指定的长度，单击“转换”按钮后，会在“格式化后的字符串”文本框中显示格式化后的数字。

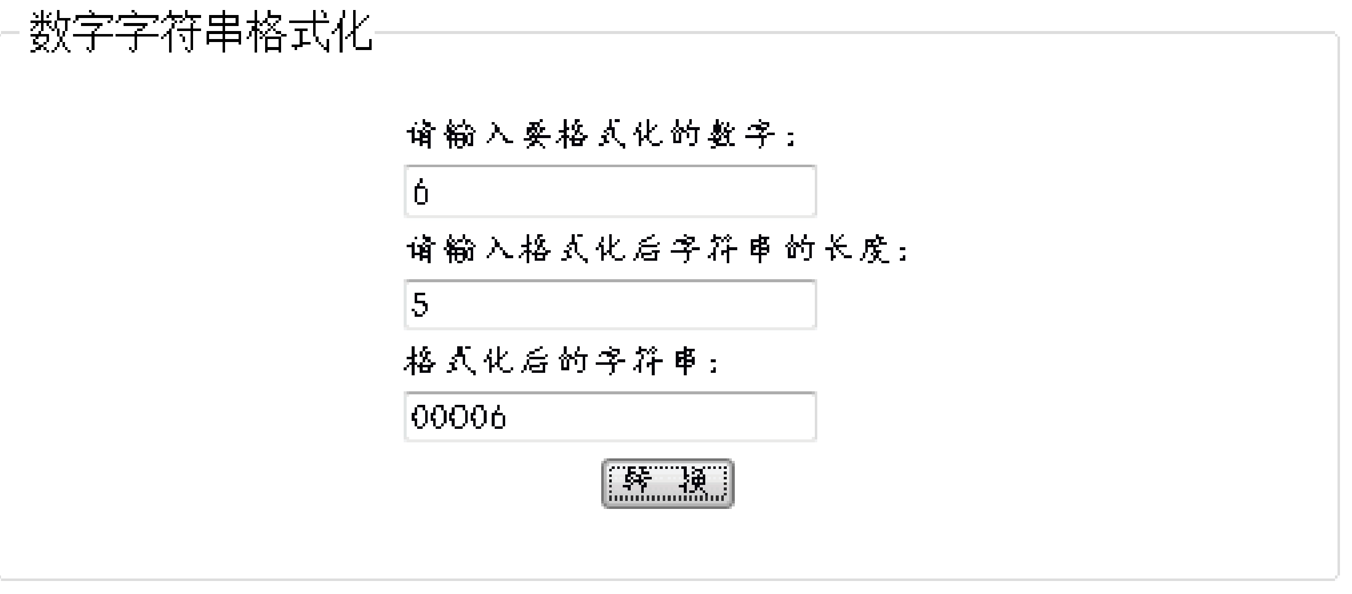


图 9.19　将数字字符串格式化为指定长度

关键技术

本实例主要是在自定义的 JavaScript 函数中通过循环来实现。首先获得要格式化的数字和格式长度，然后在循环中对要格式化的数字之前用“0”补齐，最后将这个格式化后的数字字符串返回。

设计过程

（1）新建 index.jsp 页，编写将数字字符串格式化为指定长度的 JavaScript 函数，关键代码如下：

```
/**
*格式化数字
*@number ： 要格式化的数字
*@len：格式长度
*@return ： 返回格式化后的数字
*/
function formatNum(number,len){
	var strLength = len - number.length;		//格式长度减去数字的长度，就是在数字前补 0 的个数
	for(var i=0; i<strLength;i++){
		number = "0"+number;
	}
	return number;
}
```

（2） 编写表单提交按钮 onClick 事件所调用的 JavaScript 函数，在该函数中首先验证输入是否为空和是否为数字，如果输入正确，则调用格式化数字的自定义 JavaScript 函数将数字格式化为指定长度，关键代码如下：

```
function convert(){
	var number = document.getElementById("number").value;
	var num_len = document.getElementById("num_len").value;
	if(number==""){					//验证输入的数字是否为空
		alert("请输入要格式化的数字！");
		document.getElementById("number").focus();
		return;
	}
	if(isNaN(number)){				//验证输入的是否为数字
		alert("您输入的数字不正确！");
		document.getElementById("number").focus();
		return;
	}
	if(num_len==""){				//验证输入的长度是否为空
		alert("请输入格式化后字符串的长度！");
		document.getElementById("num_len").focus();
		return;
	}
	if(isNaN(number)){				//验证输入的长度是否为数字
		alert("您输入的格式化字符串的长度不正确！");
		document.getElementById("num_len").focus();
		return;
	}
	//调用格式化数字的方法，并将返回值赋给文本框
	document.getElementById("convertStr").value = formatNum(number,num_len);
}
```

秘笈心法

根据本实例，可以在实现自动编号时使用该函数，在网页中显示指定长度的日期字符串时也需要使用该函数。

实例 225	限制 Textarea 文本域内容的长度 光盘位置：光盘\MR\09\225	初级 实用指数：★★★★

实例说明

在实际的开发过程中，经常需要限制用户在文本域中输入数据的长度，如论坛中的留言内容的长度。运行本实例，如图 9.20 所示，在文本域中输入留言内容，当留言内容超过指定字符长度时，会终止用户继续输入。

关键技术

实现本实例，首先需要通过 charCodeAt()函数获取字符的 Unicode 值，然后根据字符的 Unicode 值的范围判

断字符串中是否包含中文字符，如果包含中文字符，则将定义好的计数器的值加 2，否则加 1，最后判断计数器的值是否超过指定的字符长度范围，如果超过指定长度范围，则截取字符串。



图 9.20　限制 Textarea 文本域内容的长度

设计过程

（1）新建 index.jsp 页，编写限制文本域输入内容长度的 JavaScript 函数。在该函数中，通过循环计算出文本域内容的长度，由于中文字符是占两个字节的，所以计算时应该判断文本域内容是否存在中文字符，关键代码如下：

```
function limitTextarea(){
	var max=20;							//最多可输入 20 个字符，中文占两个字节
	var areaStr = document.getElementById("str").value;
	var temp=0;
	for(var i=0;i<areaStr.length;i++){				//可能包含中文，需要循环判断
		var code = areaStr.charCodeAt(i);			//转换为 Unicode 值
		if(code>255){						//大于 255 的为中文字符
			temp=temp+2;
		}
		else{
			temp=temp+1;
		}
		if(temp> max){						//如果字符长度超过指定长度，则跳出循环
			break;
		}
	}
	document.getElementById("str").value=areaStr.substring(0,i);
	document.getElementById("now_len").innerHTML = temp;		//当前输入的字符长度
	document.getElementById("remainder_len").innerHTML = max-temp;	//剩余字符长度
}
```

（2）在<body>标签的 onload 加载事件中调用步骤（1）中编写的函数，并在<textarea>的 onkeyup 事件和 onkeydown 事件中调用步骤（1）中的函数，实时跟踪用户输入的内容长度，然后在文本域之后加入显示“当前字数”和“剩余字数”的<span>标签，关键代码如下：

```
<body onload="limitTextarea()" >
	<textarea rows="5" cols="40" id="str"
		onkeyup="limitTextarea()" onkeydown="limitTextarea()"></textarea>
	当前字数：<span   id="now_len" /></span>  剩余字数：<span id="remainder_len" ></span>
</body>
```

秘笈心法

根据本实例，读者可以实现限制留言簿中文本域的文字内容长度，以及限制用户注册时添加备注信息的文本域的文字内容长度。

实例 226	将长数字分位显示 光盘位置：光盘\MR\09\226	初级 实用指数：★★★☆

实例说明

在显示某项统计数据结果时，这个统计结果有可能是一串长数字，为了方便用户更清晰地查看数据，需要

将长数字分位显示。运行本实例，如图 9.21 所示，在“请输入要转换的长数字”文本框中输入一个长数字，单击“转换”按钮后，会将数字以“，”隔开分位显示。

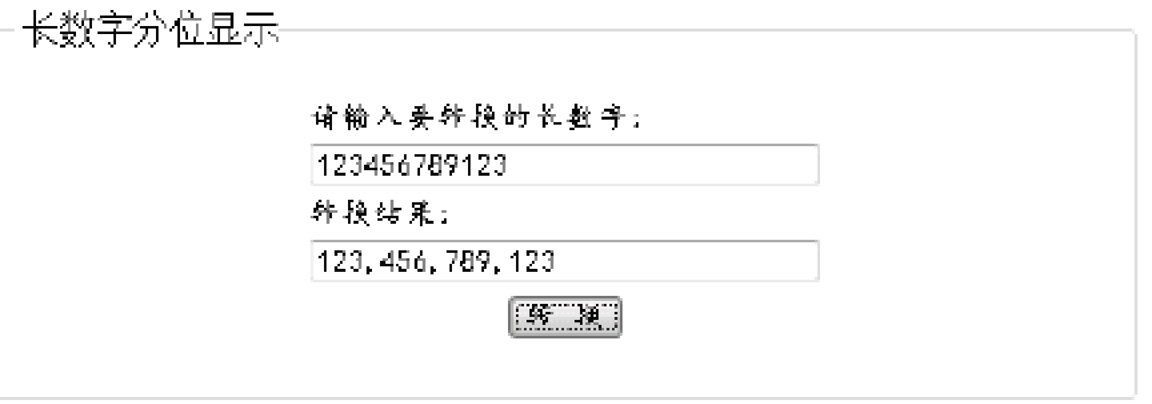


图 9.21　将长数字分位显示

关键技术

本实例主要通过循环数字字符串来实现。首先判断输入的是否为数字，如果为数字，则在循环中设置一个指定的步长，然后每隔指定的步长加一个逗号分隔开，最后组合成一个带有逗号分隔符的数字字符串。

设计过程

（1）新建 index.jsp 页，编写将数字分位的 JavaScript 函数，关键代码如下：

```
function compart(lang_num){
        var result=0;                                   //保存分位后的结果
        var dec="";                                     //保存小数点后的数字
        if(lang_num<4){                                 //如果小于 4 位，直接返回
                result = lang_num;
        }
        else{                                           //如果大于 4 位
                var decimal = lang_num.indexOf(".");    //是否包含小数
                var temp="";                            //保存整数部分的分位字符串
                var res="";                             //保存没有分位的整数
                if(decimal>0){                          //如果包含小数
                        dec=lang_num.substr(decimal);   //截取小数点后的值
                        res=lang_num.substr(0,decimal); //截取小数点前的整数值
                }else{                                  //如果不存在小数部分
                        res=lang_num;
                }
                for(var i=res.length;i>0;i=i-3){        //循环整数，从整数的个位开始循环
                        if(i-3>0){                      //每隔 3 位加一个逗号
                                temp=","+res.substr(i-3,3)+temp;
                        }
                        else{                           //少于 3 位时的情况
                                temp=res.substr(0,i)+temp
                        }
                }
                result =temp+dec;
        }
        return result;
}
```

（2）编写表单提交按钮 onClick 事件调用的 JavaScript 函数，在该函数中首先验证用户输入的数字是否为空和是否为数字，如果验证成功，则调用分位显示数字的函数，将数字分位显示在文本框中，关键代码如下：

```
function convert(){
        var lang_number = document.getElementById("lang_number").value;
        if(lang_number==""){
                alert("请输入数字！");
                document.getElementById("lang_number").focus();
                return;
        }
        if(isNaN(lang_number)){
                alert("您输入的数字无效！");
                document.getElementById("lang_number").focus();
                return;
        }
        document.getElementById("result_num").value = compart(lang_number);
}
```

秘笈心法

根据本实例，读者可以实现在房地产信息网中将产品报价分位显示。

实例 227　将 RGB 格式的颜色值转换为十六进制

光盘位置：光盘\MR\09\227

初级

实用指数：★★★★

实例说明

在网站开发过程中，经常会遇到将 RGB 格式的颜色值转换为十六进制格式的情况，例如，在实现文字动态变色时，就需要使用该方法将 RGB 值转换为十六进制格式。运行本实例，如图 9.22 所示，在相应的文本框中分别输入 R、G、B 的颜色值，单击"转换"按钮后，会在"转换后十六进制的颜色值"文本框中显示一个十六进制的 RGB 颜色值。

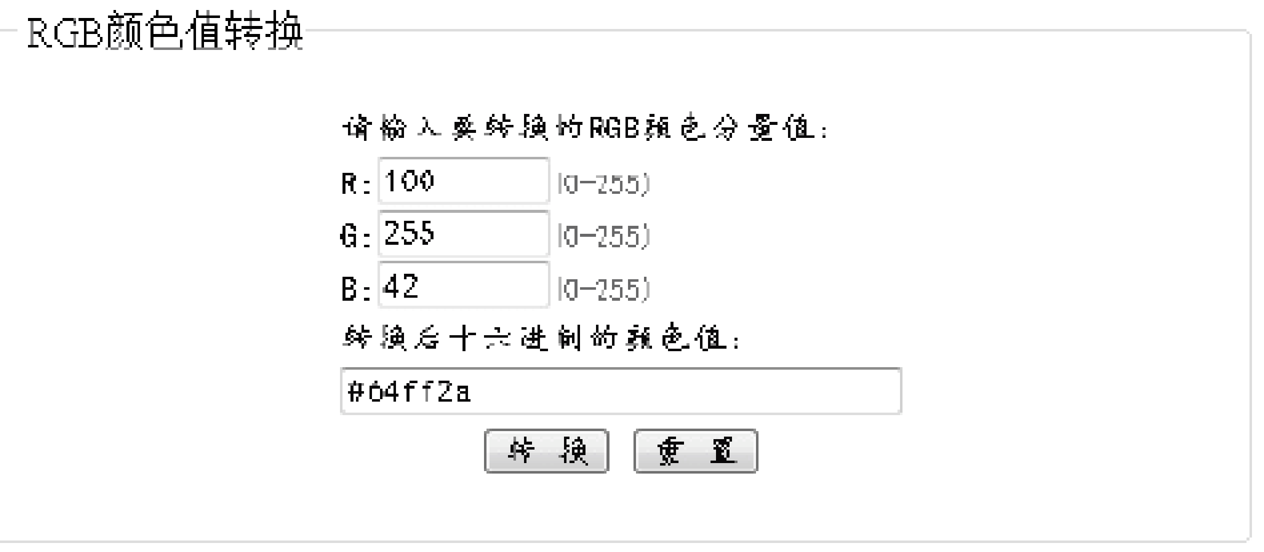


图 9.22　将 RGB 颜色值转换为十六进制

关键技术

本实例主要应用 JavaScript 中提供的 parseInt()方法和 Number 对象的 toString()方法。parseInt()方法用于返回由字符串转换得到的整数。Number 对象的 toString()方法用于将数值转换为字符串，该方法可以返回数字的不同进制的值，代码如下：

```
var num = 20;
num.toString(2);                //转换为二进制，结果为 10100
num.toString(8);                //转换为八进制，结果为 24
num.toString(10);               //转换为十进制，结果为 20
num.toString(16);               //转换为十六进制，结果为 14
```

设计过程

（1）新建 index.jsp 页，编写转换 RGB 颜色值为十六进制的 JavaScript 函数，关键代码如下：

```
function toHex(r,g,b){
//如果 R、G、B 的值为空，修改为 0
if(r==""){
        r=0;
}
if(g==""){
        g=0;
}
if(b==""){
        b=0;
}
var red = parseInt(r).toString(16);         //R 值的十六进制字符串
if(red.length<2){                           //少于 2 位，补 0
        red="0"+red;
}
var green = parseInt(g).toString(16);       //G 值的十六进制字符串
if(green.length<2){                         //少于 2 位，补 0
        green="0"+green;
}
var blue = parseInt(b).toString(16);        //B 值的十六进制字符串
```

```
if(blue.length<2){                    //少于 2 位，补 0
        blue="0"+blue;
}
return "#"+red+green+blue;            //组合成一个 RGB 颜色字符串
}
```

（2）编写按钮 onClick 事件所调用的 JavaScript 函数，在该函数中首先验证输入的 R、G、B 值是否符合要求，如果验证成功，则调用步骤（1）中定义的转换为十六进制的函数，最后将转换后的 RGB 颜色值显示在文本框中，关键代码如下：

```
function convert(){
        var R = document.getElementById("r_value").value;
        var G = document.getElementById("g_value").value;
        var B = document.getElementById("b_value").value;
        if(isNaN(R)){
                alert("您输入的(R)颜色值必须为 0～255 之间的数字！");
                return ;
        }else{
                if(R>255){
                        alert("请输入 0～255 之间的数字！");
                        return;
                }
        }
        if(isNaN(G)){
                alert("您输入的(G)颜色值必须为 0～255 之间的数字！");
                return ;
        }else{
                if(G>255){
                        alert("请输入 0～255 之间的数字！");
                        return;
                }
        }
        if(isNaN(B)){
                alert("您输入的(B)颜色值必须为 0～255 之间的数字！");
                return ;
        }else{
                if(B>255){
                        alert("请输入 0～255 之间的数字！");
                        return;
                }
        }
        document.getElementById("hex_num").value = toHex(R,G,B);
}
```

秘笈心法

根据本实例，读者可以实现彩色渐变的文字动画，还可以实现表格或页面背景循环变色。

实例 228 从指定 URL 中提取文件名

初级

光盘位置：光盘\MR\09\228

实用指数：★★★★

实例说明

在开发网站时，可能会遇到从指定的 URL 中提取文件名的情况。运行本实例，如图 9.23 所示，在“请输入 URL 地址”文本框中输入一个 URL 地址，单击“提取”按钮后，在“提取的文件名”文本框中将显示提取出的文件名。



图 9.23　提取 URL 中的文件名

关键技术

本实例主要通过 String 对象的 replace()方法实现。该方

法用于替换一个与正则表达式匹配的子串，该方法的替换字串可以包含$字符，包含$字符时具有特殊含义，具体含义如表 9.2 所示。

表 9.2　包含$字符的字符串的具体含义

字　符	含　义
$&	指定与整个模式匹配的子串
$`	位于匹配字串左侧的文本
$'	位于匹配字串右侧的文本
$n	捕获的第 n 个子匹配，此处 n 为从 1～9 的十进制一位数
$nn	捕获的第 nn 个子匹配，此处 nn 为从 01～99 的十进制两位数

设计过程

（1）新建 index.jsp 页，编写从指定 URL 中提取文件名的 JavaScript 自定义函数，关键代码如下：

```
function getFileName(url){
	var regExp = /(.*\/)*([^.]+).*/ig;
	url = url.replace(regExp,"$2");
	return url;
}
```

（2）编写表单提交按钮 onclick 事件调用的 JavaScript 函数，用于验证用户输入的 URL 地址是否有效，如果验证成功，则提取文件名并显示在相应的文本框中，关键代码如下：

```
function check(){
	var url = document.getElementById("url").value;
	//验证 URL 是否正确的正则表达式
	var urlExp = /http(s)?:\/\/([\w-]+\.)+[\w-]+(\/[\w- .\/?%&=]*)?(\/\w+)*.\w{3}$/;
	if(url==""){
		alert("请输入 URL 地址！ ");
		document.getElementById("url").focus();
		return;
	}
	if(!urlExp.test(url)){
		alert("您输入的 URL 地址无效！ ");
		document.getElementById("url").focus();
		return;
	}
	document.getElementById("filename").value = getFileName(url);
}
```

秘笈心法

String 对象的 replace()方法用于替换一个与正则表达式匹配的子串，其语法结构如下：

```
stringObj.replace(regExp,replaceText)
```

参数说明

❶ stringObj：必选参数。要执行该替换的 String 对象或字符串文字。

❷ regExp：必选参数。为包含正则表达式模式或可用标志的正则表达式对象，也可以是 String 对象或文字。如果 regExp 不是正则表达式对象，它将被转换为字符串，并进行精确的查找；不要尝试将字符串转化为正则表达式。

❸ replaceText：必选参数。一个 String 对象或字符串文字，对于 stringObj 中每个匹配 regExp 中的位置都用该对象所包含的文字加以替换。replaceText 参数也可以是返回替换文本的函数。

9.3　日期时间处理

在 Web 应用开发过程中，经常需要处理表单中输入的日期时间类型的数据，如计算两个日期相差的天数、计算某日期是星期几、实时显示系统时间等。本节将通过几个实例讲解如何通过 JavaScript 来处理日期时间。

实例 229 计算两个日期相差的天数

光盘位置：光盘\MR\09\229

初级

实用指数：★★★★

实例说明

在实际的开发过程中，可能会遇到计算两个日期相差的天数的情况。运行本实例，如图 9.24 所示，在相应的文本框中分别输入两个日期字符串（格式为 yyyy-mm-dd），单击“计算”按钮后，会显示出这两个日期相差的天数。



图 9.24　计算两个日期相差的天数

关键技术

本实例主要利用 JavaScript 中的 Date 对象来实现。该对象是一个有关日期和时间的对象，创建 Date 对象的语法结构如下：

```
dateObj = new Date()
dateObj = new Date(dateVal)
dateObj = new Date(year, month, date[, hours[, minutes[, seconds[,ms]]]])
```

参数说明

❶ dateObj：要赋值为 Date 对象的变量名。

❷ dateVal：如果是数字值，dateVal 表示指定日期与 1970 年 1 月 1 日午夜间全球标准时间的毫秒数。如果是字符串，则 dateVal 按照 parse 方法中的规则进行解析。dateVal 参数也可以是从某些 ActiveX(R)对象返回的 VT_DATE 值。

❸ year：表示完整的年份（yyyy）。

❹ month：表示月份，从 0～11 之间的整数（1 月～12 月）。

❺ date：表示日期，从 1～31 之间的整数。

❻ hours：可选参数。表示小时，从 0～23 之间的整数。

❼ minutes：可选参数。表示分钟，从 0～59 之间的整数。

❽ seconds：可选参数。表示秒，从 0～59 之间的整数。

❾ ms：可选参数。表示毫秒，从 0～999 之间的整数。

Date 对象提供了获取和设置有关日期和时间的方法，如表 9.3 所示。

表 9.3　Date 对象的方法及说明

方　法	说　明
getFullYear()/setFullYear()	返回/设置 Date 对象中的完整年份值（yyyy）
getMonth()/setMonth()	返回/设置 Date 对象中的月份值（0～11 的整数）
getDate()/setDate()	返回/设置 Date 对象中表示的一个月中的日期值
getHours()/setHours()	返回/设置 Date 对象中的小时值
getMinutes()/setMinutes()	返回/设置 Date 对象中的分钟值
getSeconds()/setSeconds()	返回/设置 Date 对象中的秒值
getMilliseconds()/setMilliseconds()	返回/设置 Date 对象中的毫秒值
getTime()	返回 Date 对象中的时间值，该方法返回一个整数值，这个整数代表了从 1970 年 1 月 1 日开始计算到 Date 对象中的时间之间的毫秒数
setTime()	设置 Date 对象中的日期和时间值
getDay()	返回 Date 对象中的一周中表示的日期值，星期天到星期六的值是从 0～6

设计过程

（1）新建 index.jsp 页，编写计算两个日期之间相差的天数的 JavaScript 自定义函数，在该函数中，首先根据日期类型的字符串（yyyy-mm-dd）创建日期类型的 Date 对象，然后通过 Date 对象的 getTime()方法获得表示日期的毫秒值，通过两个日期的毫秒值之差除以一天的毫秒值即两个日期相差的天数，关键代码如下：

```
/**
*计算两个日期相差的天数
*@date1：日期类型的字符串（yyyy-mm-dd）
*@date2：日期类型的字符串 （yyyy-mm-dd）
*@return：返回日期天数差
*/
function getDays(date1,date2){
	var date1Str = date1.split("-");			//将日期字符串分隔为数组，数组元素分别为年、月、日
	//根据年、月、日的值创建 Date 对象
	var date1Obj = new Date(date1Str[0],(date1Str[1]-1),date1Str[2]);
	var date2Str = date2.split("-");
	var date2Obj = new Date(date2Str[0],(date2Str[1]-1),date2Str[2]);
	var t1 = date1Obj.getTime();			//返回从 1970-1-1 开始计算到 Date 对象中的时间之间的毫秒数
	var t2 = date2Obj.getTime();			//返回从 1970-1-1 开始计算到 Date 对象中的时间之间的毫秒数
	var datetime=1000*60*60*24;			//表示一天 24 小时时间内的毫秒值
	var minusDays = Math.floor(((t2-t1)/datetime));	//计算出两个日期天数差
	var days = Math.abs(minusDays);			//如果结果为负数，取绝对值
	return days;
}
```

（2）编写表单提交按钮 onClick 事件所调用的 JavaScript 函数，在该函数中首先验证输入的日期是否为空，如果不为空，需要验证日期是否有效。如果验证成功，则调用步骤（1）编写的计算两个日期相差天数的函数，最后将返回结果赋值给相应的文本框，关键代码如下：

```
function check(){
	var start_date = document.getElementById("start_date").value;
	var end_date = document.getElementById("end_date").value;
	if(start_date==""){
		alert("请输入开始日期！");
		return;
	}
	else{
		if(!checkDate(start_date)){
			alert("您输入的开始日期无效！");
			return;
		}
	}
	if(end_date==""){
		alert("请输入终止日期！");
		return;
	}else{
		if(!checkDate(end_date)){
			alert("您输入的终止日期无效！");
			return;
		}
	}
	document.getElementById("minusDay").value = getDays(start_date,end_date);
}
```

秘笈心法

本实例在实现时，首先通过 Date 对象的 getTime()方法获得两个日期对象的毫秒值，然后再通过两个日期的毫秒值之差除以表示一天的毫秒值（1000×24×60×60），即两个日期相差的天数。由于计算结果可能不是整数，因此需要通过 Math.floor()函数取整。

实例 230　计算两个日期相差的小时数

初级

光盘位置：光盘\MR\09\230

实用指数：★★★★

实例说明

在实际开发应用中，经常需要计算两个日期相差的小时数。运行本实例，如图 9.25 所示，在相应的文本框中分别输入开始日期和终止日期，单击“计算”按钮后，会在文本框中显示这两个日期之间相差的小时数。

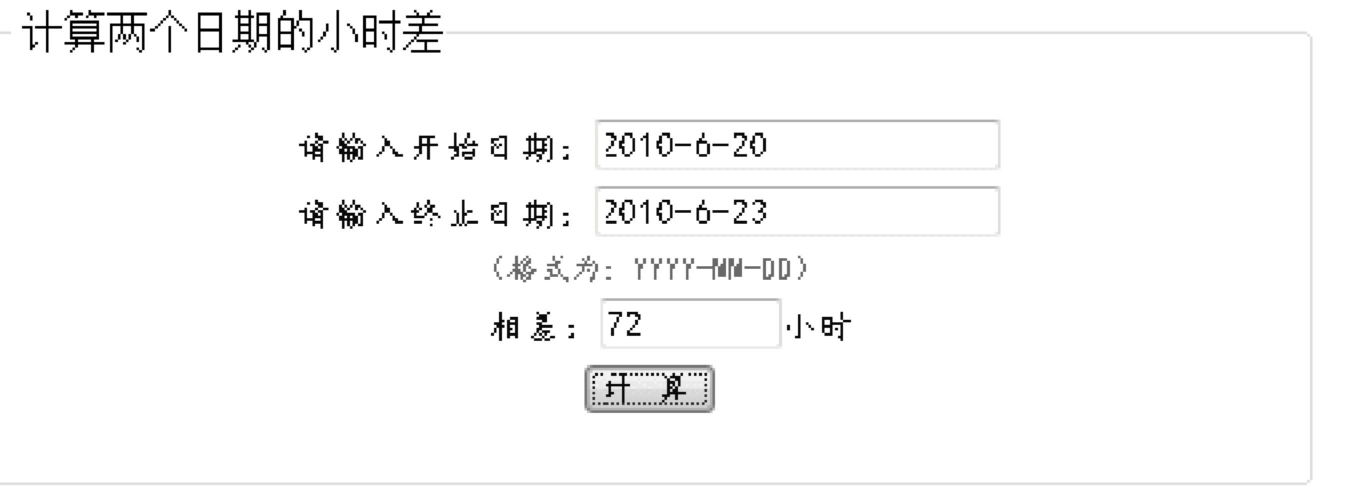


图 9.25　计算两个日期之间的小时差

关键技术

实现本实例很简单，首先需要计算出两个日期之间相差的天数，然后再乘以表示一天的 24 小时就是两个日期相差的小时数。

设计过程

（1）新建 index.jsp 页，编写计算两个日期之间相差的小时数的 JavaScript 函数，关键代码如下：

```
/**
 *计算两个日期相差的小时数
*@date1：日期类型的字符串（yyyy-mm-dd）
*@date2：日期类型的字符串 （yyyy-mm-dd）
 *@return：返回两个日期相差的小时数
 */
function getMinusHours(date1,date2){
	var date1Str = date1.split("-");			//将日期字符串分隔为数组，数组元素分别为年、月、日
	//根据年、月、日的值创建 Date 对象
	var date1Obj = new Date(date1Str[0],(date1Str[1]-1),date1Str[2]);
	var date2Str = date2.split("-");
	var date2Obj = new Date(date2Str[0],(date2Str[1]-1),date2Str[2]);
	var t1 = date1Obj.getTime();				//返回从 1970-1-1 开始计算到 Date 对象中的时间之间的毫秒数
	var t2 = date2Obj.getTime();				//返回从 1970-1-1 开始计算到 Date 对象中的时间之间的毫秒数
	var datetime=1000*60*60*24;				//一天时间的毫秒值
	var minusDays = Math.floor(((t2-t1)/datetime));	//计算出两个日期天数差
	var days = Math.abs(minusDays);			//如果结果为负数，取绝对值
	var minusHours = days*24;
	return minusHours;
}
```

（2）编写按钮 onClick 事件调用的 JavaScript 函数。首先验证输入的日期，验证时用到了 checkDate()函数，如果验证成功，则调用计算两个日期相差小时数的函数，最后将返回的小时数赋给相应的文本框，关键代码如下：

```
function check(){
	var start_date = document.getElementById("start_date").value;
	var end_date = document.getElementById("end_date").value;
	if(start_date==""){
		alert("请输入开始日期！");
		return;
	}
	else{
		if(!checkDate(start_date)){
			alert("您输入的开始日期无效！");
			return;
		}
	}
	if(end_date==""){
		alert("请输入终止日期！");
		return;
	}else{
```

```
            if(!checkDate(end_date)){
                alert("您输入的终止日期无效！");
                return;
            }
        }
        document.getElementById("minus_hour").value = getMinusHours(start_date,end_date);
}
```

秘笈心法

本实例在实现时，首先通过 Date 对象的 getTime()方法获得两个日期对象的毫秒值，然后再通过两个日期的毫秒值之差除以表示一天的毫秒值（1000×24×60×60）计算出两个日期相差的天数，最后将相差的天数乘以 24 得到两个日期相差的小时数。

实例 231　计算某一天是星期几

光盘位置：光盘\MR\09\231

初级　实用指数：★★★★

实例说明

本实例将介绍如何计算某一日期是星期几。运行本实例，如图 9.26 所示，在文本框中输入日期类型的字符串，单击“计算”按钮后，会显示出日期所对应的星期值。

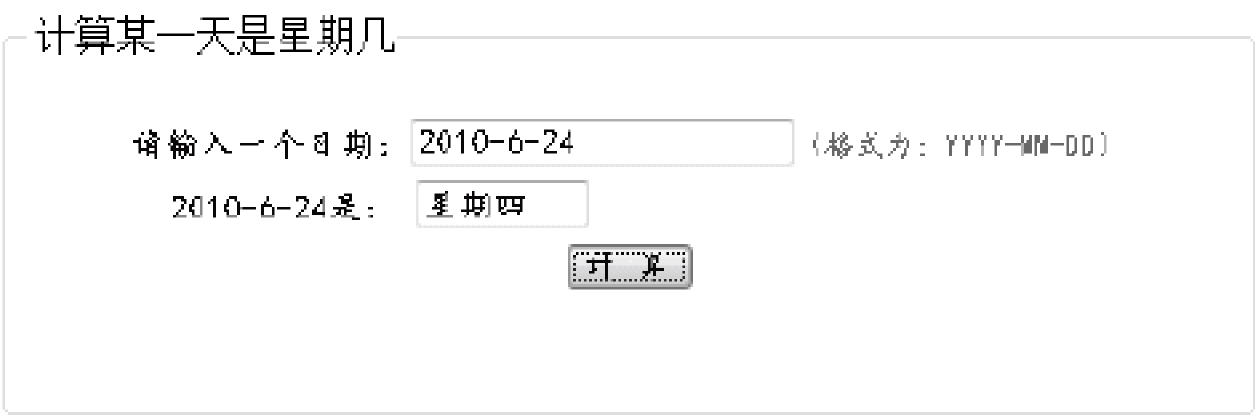


图 9.26　计算某一天是星期几

关键技术

本实例的实现主要是利用 Date 对象中的 getDay()方法。该方法返回的是 Date 对象中一周的星期值，星期值的范围是 0～6 的整数，星期日为 0、星期一为 1、星期二为 2，依此类推。

设计过程

（1）新建 index.jsp 页，编写计算某一日期是星期几的 JavaScript 函数，关键代码如下：

```
/**
 *计算某一天是星期几
 *@date：日期类型的字符串（yyyy-mm-dd）
 *@return：返回星期值
 */
function getWeekByDate(date){
        var dateStr = date.split("-");//将日期字符串分隔为数组，数组元素分别为年、月、日
        //根据年、月、日的值创建 Date 对象
        var dateObj = new Date(dateStr[0],(dateStr[1]-1),dateStr[2]);
        var weeks = ["星期日","星期一","星期二","星期三","星期四","星期五","星期六"];
        return weeks[dateObj.getDay()];
}
```

（2）编写按钮 onClick 事件所调用的 JavaScript 函数，在该函数中调用计算某一天是星期几的函数，然后将获得的星期值赋给一个文本框，关键代码如下：

```
function check(){
        var date = document.getElementById("date").value;
        if(date==""){
            alert("请输入日期！");
            return;
        }
        else{
            if(!checkDate(date)){
                alert("您输入的日期无效！");
                return;
```

```
            }
        }
        document.getElementById("dateStr").innerHTML="  "+date+"是：";
        document.getElementById("week").value = getWeekByDate(date);
}
```

秘笈心法

创建一个 Date 对象时，月份的值应该是从 0～11 之间的整数，如果月份的值设置错误，使用 getDay()方法返回的星期值肯定是不对的。如根据日期字符串“2010-5-5”来创建一个 Date 对象，正确的创建方法代码如下：

```
var dateObj = new Date(2010,4,5);//表示的日期为 2010-5-5
```

实例 232 显示长日期格式的系统时间

光盘位置：光盘\MR\09\232

初级

实用指数：★★★★

实例说明

所谓长日期格式就是以“YYYY 年 MM 月 DD 日 星期 W”格式来显示的。在网站的适当位置加入此格式的日期不仅可以美化界面，而且还可以达到方便用户的目的。运行本实例，如图 9.27 所示，在网页中会以长日期格式显示当前的系统日期。

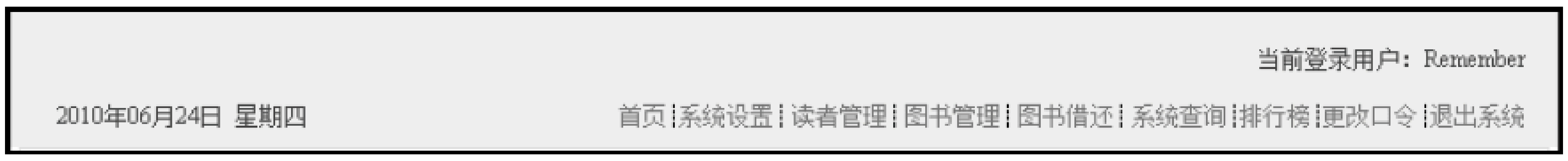


图 9.27 显示长日期格式的系统时间

关键技术

本实例主要是利用 JavaScript 中的 Date 对象来实现的。创建一个 Date 对象时，如果该对象不包含任何构造参数，并且不通过 setXXX()方法设置该对象中的年、月、日等值，那么该对象则表示当前系统时间的 Date 对象，因此可以根据 Date 对象的 getXXX()方法来获得当前系统时间的年、月、日、小时、分、秒等值。

设计过程

（1）新建 index.jsp 页，编写获得当前系统日期的长日期格式的 JavaScript 函数。在该函数中首先创建一个表示当前系统时间的 Date 对象，然后获得该对象中的年、月、日及星期值组合成一个字符串，关键代码如下：

```
/**
*显示长日期格式的系统日期
*/
function getLangDate(){
        var dateObj = new Date();              //表示当前系统时间的 Date 对象
        var year = dateObj.getFullYear();      //当前系统时间的完整年份值
        var month = dateObj.getMonth()+1;      //当前系统时间的月份值
        var date = dateObj.getDate();          //当前系统时间的月份中的日
        var day = dateObj.getDay();            //当前系统时间中的星期值
        var weeks = ["星期日","星期一","星期二","星期三","星期四","星期五","星期六"];
        var week = weeks[day];                 //根据星期值，从数组中获取对应的星期字符串
        if(month<10){
                month="0"+month;
        }
        if(date<10){
                date="0"+date;
        }
        var newDate = year+"年"+month+"月"+date+"日    "+week;
        document.getElementById("dateStr").innerHTML = newDate;
}
```

（2）要想在页面加载后显示系统时间，需要通过页面<body>标签的 onload 事件来调用步骤（1）中编写的

JavaScript 函数，并在页面中设置一个 id 为 dateStr 的<div>标签，关键代码如下：

```
<body onLoad="getLangDate()">
        <div id="dateStr" class="word_Green"></div>
</body>
```

秘笈心法

本实例在从 Date 对象中获得年份时，使用的是 getFullYear()方法而不是 getYear()方法。getFullYear()方法以绝对数字的形式返回年份值。例如，1986 年的返回值就是 1986。这样可以避免出现 2000 年问题，从而不会将 2000 年 1 月 1 日以后的日期与 1900 年 1 月 1 日以后的日期混淆。

实例 233　实时显示系统时间

光盘位置：光盘\MR\09\233

初级

实用指数：★★★★

实例说明

在浏览很多网站时，经常会发现在网站中加入了显示当前系统时间的功能，在网页中显示当前系统时间，不仅可以方便浏览者掌握当前时间，而且还美化了网页。运行本实例，如图 9.28 所示，在网页的导航条下方实时显示了当前的系统时间。



图 9.28　实时显示系统时间

关键技术

本实例主要是通过利用 Date 对象来实现的。首先创建一个表示当前系统时间的 Date()对象，然后通过该对象的 getXXX()方法获得当前系统时间的年、月、日、小时、分、秒和星期值，接下来将获得的这些值组合成一个日期时间字符串，并将日期时间字符串设置成为<div>标签的内容，最后通过 window 对象的 setTimeout()函数每隔 1 秒调用一个实时显示系统时间的函数。

设计过程

（1）新建 index.jsp 页，编写实时显示系统时间的 JavaScript 函数，关键代码如下：

```
/**
*实时显示系统时间
*/
function getLangDate(){
        var dateObj = new Date();                //表示当前系统时间的 Date 对象
        var year = dateObj.getFullYear();        //当前系统时间的完整年份值
        var month = dateObj.getMonth()+1;        //当前系统时间的月份值
        var date = dateObj.getDate();            //当前系统时间的月份中的日
        var day = dateObj.getDay();              //当前系统时间中的星期值
        var weeks = ["星期日","星期一","星期二","星期三","星期四","星期五","星期六"];
        var week = weeks[day];                   //根据星期值，从数组中获取对应的星期字符串
        var hour = dateObj.getHours();           //当前系统时间的小时值
        var minute = dateObj.getMinutes();       //当前系统时间的分钟值
        var second = dateObj.getSeconds();       //当前系统时间的秒钟值
        //如果月、日、小时、分、秒的值小于 10，在前面补 0
        if(month<10){
                month = "0"+month;
        }
        if(date<10){
                date = "0"+date;
        }
        if(hour<10){
                hour = "0"+hour;
        }
```

```
        if(minute<10){
                minute = "0"+minute;
        }
        if(second<10){
                second = "0"+second;
        }
        var newDate = year+"年"+month+"月"+date+"日    "+week+" "+hour+":"+minute+":"+second;
        document.getElementById("dateStr").innerHTML = "系统公告：[ "+newDate+" ]";
        setTimeout("getLangDate()",1000);//每隔 1 秒重新调用一次该函数
}
```

（2）在页面<body>标签中通过 onload 事件加载步骤（1）中编写的 JavaScript 函数，并在页面的适当位置加入<div>标签，id 为 dateStr，关键代码如下：

```
<body onLoad="getLangDate()">
        <div id="dateStr" class="word_grey"></div>
</body>
```

秘笈心法

在实现实时显示系统时间时，还可以使用 window 对象的 setIntervar()方法，该方法类似于 setTimeout()方法。不同之处是调用 window 对象的 setIntervar()方法后，会一直执行 setIntervar()方法所调用的 JavaScript 方法，而 setTimeout()方法只能被执行一次。如果要通过 setTimeout()方法一直执行某个 JavaScript 方法，setTimeout()必须写在被调用的 JavaScript 方法体内。

实例 234	倒计时 光盘位置：光盘\MR\09\234	初级 实用指数：★★★★

实例说明

在浏览一些网站时，网站中经常会根据某一天发生的重要事件给出倒计时天数，如“距北京奥运会开幕还有 30 天！”“距上海世博会开幕还有 10 天！”等。运行本实例，如图 9.29 所示，会在网页中显示距 2010 年南非世界杯开幕还有多少天的提示信息。

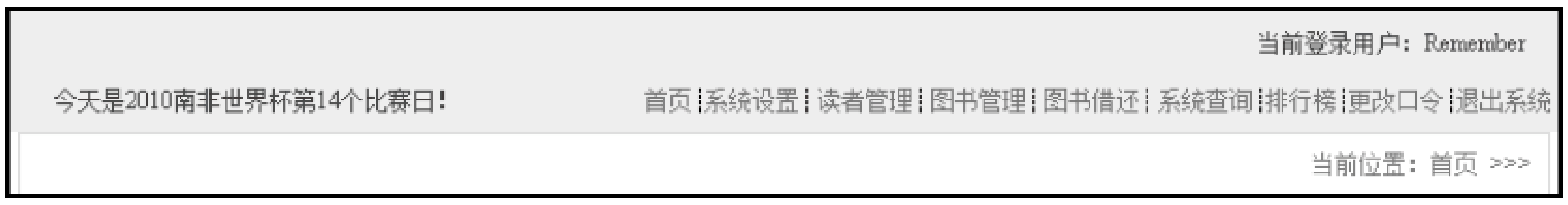


图 9.29　倒计时

关键技术

本实例主要是通过利用 JavaScript 中的 Date 对象来实现的。主要就是计算当前系统日期与某一日期之间的天数差，然后根据天数差显示相应的提示信息。计算出的天数差可能为一个浮点值，需要通过 Math 对象的 floor()方法获取整数值，如果天数差为负数，还需要使用 Math 对象取绝对值的方法 abs()。

设计过程

（1）新建 index.jsp 页，编写用于倒计时天数的 JavaScript 函数，关键代码如下：

```
/**
 *事件倒计时
 *@title：事件的名称
 *@eventDate：事件的日期
 */
function countDown(title,eventDate){
        var dateObj = new Date();                          //当前系统时间的 Date 对象
        var dateStr = eventDate.split("-");
        var eventDateObj = new Date(dateStr[0],(dateStr[1]-1),dateStr[2]);
        var t1 = dateObj.getTime();                        //获得 Date  对象中距 1970 年的时间的毫秒数
```

```
        var t2 = eventDateObj.getTime();            //获得 Date 对象中距 1970 年的时间的毫秒数
        var datetime=24*60*60*1000;                 //一天的毫秒值
        var days = Math.floor((t2-t1)/datetime)+1;  //相差的天数
        if(days>0){
                document.getElementById("day_str").innerHTML="距"+title+"开幕还有"+days+"天！";
        }
        if(days==0){
                document.getElementById("day_str").innerHTML="今天是"+title+"开幕日！";
        }
        if(days<0){
                days =Math.abs(days)+1;
                document.getElementById("day_str").innerHTML="今天是"+title+"第"+days+"个比赛日！";
        }
        //每隔一天的时间调用一次本函数，刷新显示的倒计时提示信息
        setTimeout("countDown('2010 南非世界杯','2010-6-11')",datetime);
}
```

（2）通过<body>标签的 onload 事件加载步骤（1）中编写的 JavaScript 函数，并且在页面的相应位置加入<div>标签用于显示倒计时的提示信息，关键代码如下：

```
<body onLoad="javascript:countDown('2010 南非世界杯','2010-6-11')">
        <div id="day_str" class="word_Green"></div>
</body>
```

■ 秘笈心法

根据本实例，读者可以在页面的指定位置显示一个节日倒计时，还可以在页面显示生日提醒器等。

9.4　使用 JavaScript 控制 DOM

DOM 是 Document Object Model（文档对象模型）的简称，是表示文档（如 HTML 文档）和访问、操作构成文档的各种元素（如 HTML 标记和文本串）的应用程序接口（API）。它提供了文档中独立元素的结构化、面向对象的表示方法，并允许通过对象的属性和方法访问这些对象。另外，文档对象模型还提供了添加和删除文档对象的方法，这样能够创建动态的文档内容。DOM 技术在进行 Ajax 开发时非常有用。本节将通过几个实例介绍如何应用 JavaScript 控制 DOM。

实例 235	创建节点 光盘位置：光盘\MR\09\235	初级 实用指数：★★★★

■ 实例说明

在 DOM 中，文档的层次结构被表示为一棵倒立的树，树根在上，枝叶在下，树的节点表示文档中的内容。DOM 树的根节点是个 Document 对象。本实例将介绍如何应用 JavaScript 的 Document 对象创建 HTML 元素。运行本实例，如图 9.30 所示，当单击“创建”按钮后，在网页中将创建一个 HTML 文本框元素。

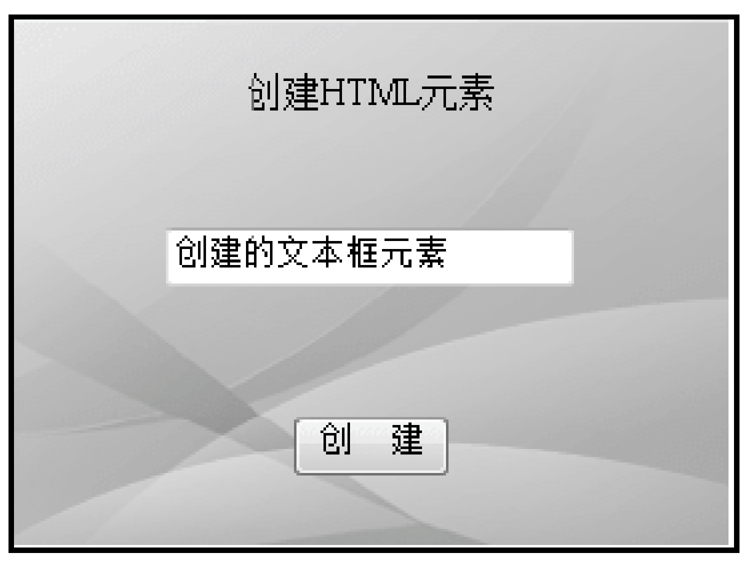


图 9.30　创建 HTML 元素

■ 关键技术

创建 HTML 元素，也就是创建一个节点，通常借助于 Document 对象的 createElement()方法来实现。createElement()方法的语法结构如下：

```
document.createElement(String elementName)
```

参数说明

❶ document：文档对象模型。

❷ elementName：表示要创建的 HTML 元素的名称。

设计过程

（1）编写创建 HTML 文本框的 JavaScript 函数，关键代码如下：

```
<script type="text/javascript">
        function createInput(){
                var element = document.createElement("input");
                element.value="创建的文本框元素";
                document.getElementById("test").appendChild(element);
        }
</script>
```

（2）创建 index.jsp 页，在按钮处的 onClick 事件中调用 JavaScript 函数，然后在表格的 id 为 test 的单元格中创建一个文本框，关键代码如下：

```
  <body>
    <table background="bg.bmp" width="270" height="200">
      <tr><td align="center">创建 HTML 元素</td> </tr>
      <tr><td id="test" align="center"></td></tr>
      <tr>
              <td align="center"><input type="button" value="创    建" onclick="createInput()"/></td>
      </tr>
    </table>
  </body>
```

秘笈心法

需要注意的是，在应用 createElement()方法创建元素时，里面的字符串并不是随意填写的，而是 HTML 文档中的一个标记名。例如，创建一个层，该参数值必须为 div；创建一个表格，该参数则必须为 table。

实例 236	添加节点 光盘位置：光盘\MR\09\236	初级 实用指数：★★★★

实例说明

当一个节点创建成功之后，一定要将该节点添加到文档中才会显示出来，本实例将介绍如何添加节点。运行本实例，如图 9.31 所示，当单击“添加”按钮后，在网页中将添加一个 HTML 层元素。



图 9.31　添加节点

关键技术

添加节点的方法有很多，对于最普遍的 HTML 元素，可以采用 appendChild()和 insertBofore()两个方法添加节点，下面分别对其进行介绍。

（1）appendChild(Node newNode)

参数说明

newNode：表示要添加的节点对象。该节点对象也就是 HTML 元素对象，如用 createElement()方法创建的节点。

（2）insertBofore(Node newNode,Node refNode)

参数说明

❶ newNode：表示要添加的节点对象。

❷ refNode：表示在该节点前插入一个节点。

设计过程

（1）编写创建 HTML 文本框的 JavaScript 函数，关键代码如下：

```
<script type="text/javascript">
        function createDiv(){
                var element = document.createElement("div");
                element.innerHTML="这就是添加的节点";
                document.getElementById("test").appendChild(element);
        }
</script>
```

（2）创建 index.jsp 页，在按钮处的 onClick 事件中调用 JavaScript 函数，然后在表格的 id 为 test 的单元格中创建一个文本框，关键代码如下：

```
<body>
  <table background="bg.bmp" width="270" height="200">
    <tr><td align="center">添加节点</td>   </tr>
    <tr><td id="test" align="center"></td></tr>
    <tr>
            <td align="center"><input type="button" value="添  加" onclick="createDiv()"/></td>
    </tr>
  </table>
</body>
```

秘笈心法

对于其他特殊的 HTML 元素，则包含了更多的添加节点的方法。例如，对于 SELECT，有更简单的方法来添加子节点；对于 TABLE 和 TR，也有其他方法添加子节点。

实例 237　为下拉列表增加选项

光盘位置：光盘\MR\09\237

初级

实用指数：★★★★

实例说明

本实例将介绍如何应用 JavaScript 的 document 对象为下拉列表添加选项。运行本实例，如图 9.32 所示，在 JavaScript 中，应用循环向下拉列表中动态添加选项。

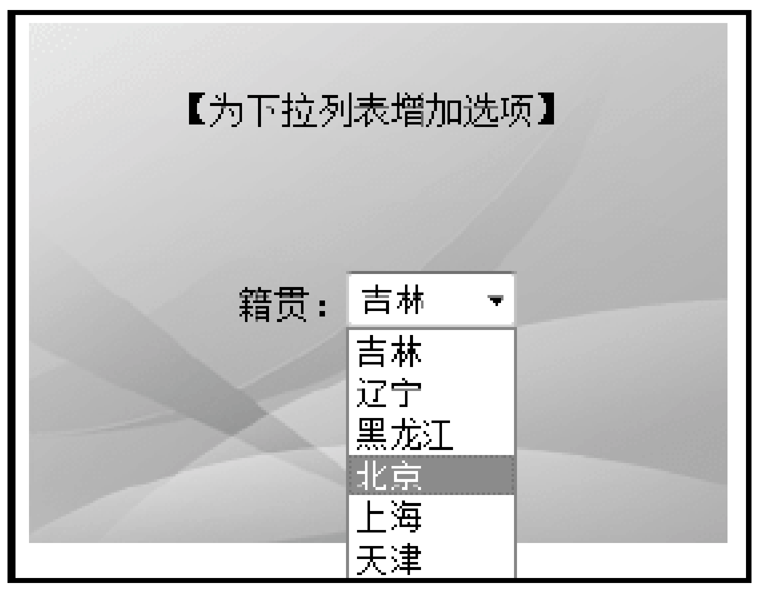


图 9.32　为下拉列表增加选项

关键技术

在应用 createElement()方法创建下拉列表元素时，将参数设置为 select 即可，然后再创建参数名为 option 的下拉列表选项，并应用创建的 SELECT 对象的 appendChild()方法将 OPTION 选项添加到下拉列表中。

设计过程

编写为下拉列表增加选项的 JavaScript 函数，关键代码如下：

```
<script type="text/javascript">
        function createSelect(){
                var selectObj = document.createElement("select");
                var str = ["吉林","辽宁","黑龙江","北京","上海","天津","河南","河北","山东","山西","江苏","安徽","云南"];
                for(var i=0;i<10;i++){
                        var op = document.createElement("option");
                        op.innerHTML=str[i];
                        selectObj.appendChild(op);
                }
                document.getElementById("test").appendChild(selectObj);
        }
</script>
```

秘笈心法

在 IE 6 之前，可以应用 add(HTMLOptionElement element,HTMLOptionElement before)方法为下拉列表添加一个选项。但是在 IE 7 以后，这个方法就不能用了，所以，为了与浏览器兼容，建议改用 appendChild()方法。

实例 238 删除下拉列表的选项

光盘位置：光盘\MR\09\238

初级

实用指数：★★★★

实例说明

本实例将介绍如何应用 JavaScript 的 document 对象删除下拉列表的选项。运行本实例，如图 9.33 所示，当单击“删除”按钮时，下拉列表的最后一个选项将被删除。

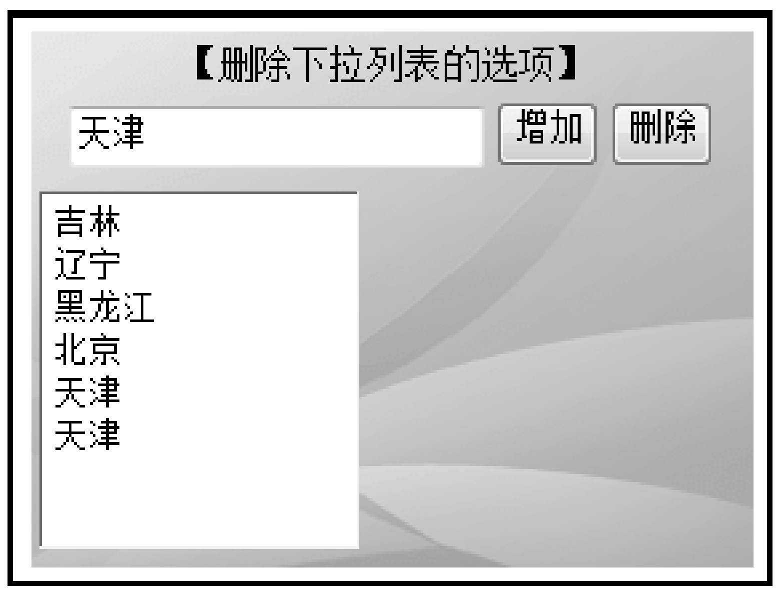


图 9.33　删除下拉列表的选项

关键技术

对于 SELECT 元素，应用的是 remove()方法来删除其中的选项，该方法的语法结构如下：

```
remove(long index)
```

参数说明

index：表示要删除的选项的索引值，如果该索引值比下拉列表中的选项的最大值还大或者小于 0，那么该方法不会删除任何选项。

设计过程

（1）编写增加下拉列表选项的 JavaScript 函数 add()，关键代码如下：

```
function add(){
	var selectObj = document.getElementById("show");
	var op = document.createElement("option");
	op.innerHTML = document.getElementById("opValue").value;
	selectObj.appendChild(op);
}
```

（2）编写删除下拉列表选项的 JavaScript 函数 del()，关键代码如下：

```
function del(){
	var selectObj = document.getElementById("show");
	selectObj.remove(selectObj.length-1);
}
```

（3）在网页的合适位置添加一个下拉列表元素，id 为 show，关键代码如下：

```
<select id="show" size="8" style="width:120px">
```

秘笈心法

在应用 document 对象的 getElementById()方法获取到 SELECT 对象之后，其 length 属性表示下拉列表中所有选项的长度。

实例 239 可编辑表格

光盘位置：光盘\MR\09\239

初级

实用指数：★★★★

实例说明

本实例将介绍如何应用 JavaScript 的 document 对象在网页中添加一个表格，而且这个表格的单元格内容是可编辑的，类似于 Excel 电子表格。运行本实例，如图 9.34 所示，双击表格中的单元格，即可对其内容进行编辑。

关键技术

本实例实现的关键之处是，每次用户双击单元格之后，单元格的值将被设置为空，并在单元格内即时插入

一个文本框，用于接收用户输入。当用户在单元格的文本框内输入相应的值且文本框失去焦点时，将单元格的文本框清空，并将文本框的值作为当前单元格的值显示出来，整个过程借助的是动态修改 HTML 元素的属性。



图 9.34　可编辑表格

在 JavaScript 中，大家都知道鼠标单击事件触发的是 onClick，而鼠标双击事件触发的是 ondblclick 事件，HTML 元素失去焦点则触发的是 onblur 事件。知道了该用哪个事件之后，再编写触发事件所调用的 JavaScript 函数即可。

设计过程

（1）创建 index.jsp 页，在该页中添加一个表格。

（2）编写动态编辑表格内容的 JavaScript 函数 edit()和元素失去焦点所调用的函数 end()，关键代码如下：

```
<script type="text/javascript">
        //双击单元格后，在单元格中创建文本框
        var inputObj = document.createElement("input");
        inputObj.type="text";
        var curCell;                                    //双击单元格后的当前单元格
        function edit(event){
                if(event==null){
                        curCell = window.event.srcElement;
                }
                else{
                        curCell = event.target;
                }
                inputObj.value=curCell.innerHTML;       //将单元格的值填充到文本框
                inputObj.onblur=end;                    //当文本框失去焦点时触发 end()函数
                curCell.innerHTML=" ";                  //清空当前单元格内容
                curCell.appendChild(inputObj);          //将文本框添加到当前单元格内
        }
        function end(){
                curCell.innerHTML=inputObj.value;
        }
</script>
```

（3）在 index.jsp 页的表格中，在鼠标双击事件 ondblclick 中调用 edit()函数，关键代码如下：

```
<table ondblclick="edit()"   border="1">
        ……
</table>
```

秘笈心法

在自定义的 edit()函数中，传递了一个表示当前事件的 event 对象，这个 event 对象是浏览器自动创建的，并将该对象传递给事件处理函数。而 event 对象的 target 属性返回的是引发事件的目标对象。例如，在表格的某个单元格中触发了 ondblclick 事件，则 event 对象的 target 返回的就是当前这个单元格对象。

第10章

Ajax 技术

- 定时业务
- 改善用户体验
- 动态加载数据

10.1　定 时 业 务

在 Web 应用开发过程中，经常会用到一些定时业务。例如，考试计时并自动提交试卷、定时保存草稿等。下面将通过这两个例子具体介绍如何实现定时业务。

实例 240　考试计时并自动提交试卷　高级

光盘位置：光盘\MR\10\240　实用指数：★★★☆

实例说明

在开发网络考试系统时，考试计时并自动提交试卷是必不可少的功能。由于在答卷过程中，试卷不能刷新，所以需要使用 Ajax 实现无刷新操作。运行本实例，访问准备考试的页面 index.jsp，在该页面中，单击“开始考试”按钮，将打开新窗口显示开始考试的页面，如图 10.1 所示，页面会自动计时，当考试时间结束时，将自动提交试卷，并弹出如图 10.2 所示的提示对话框。

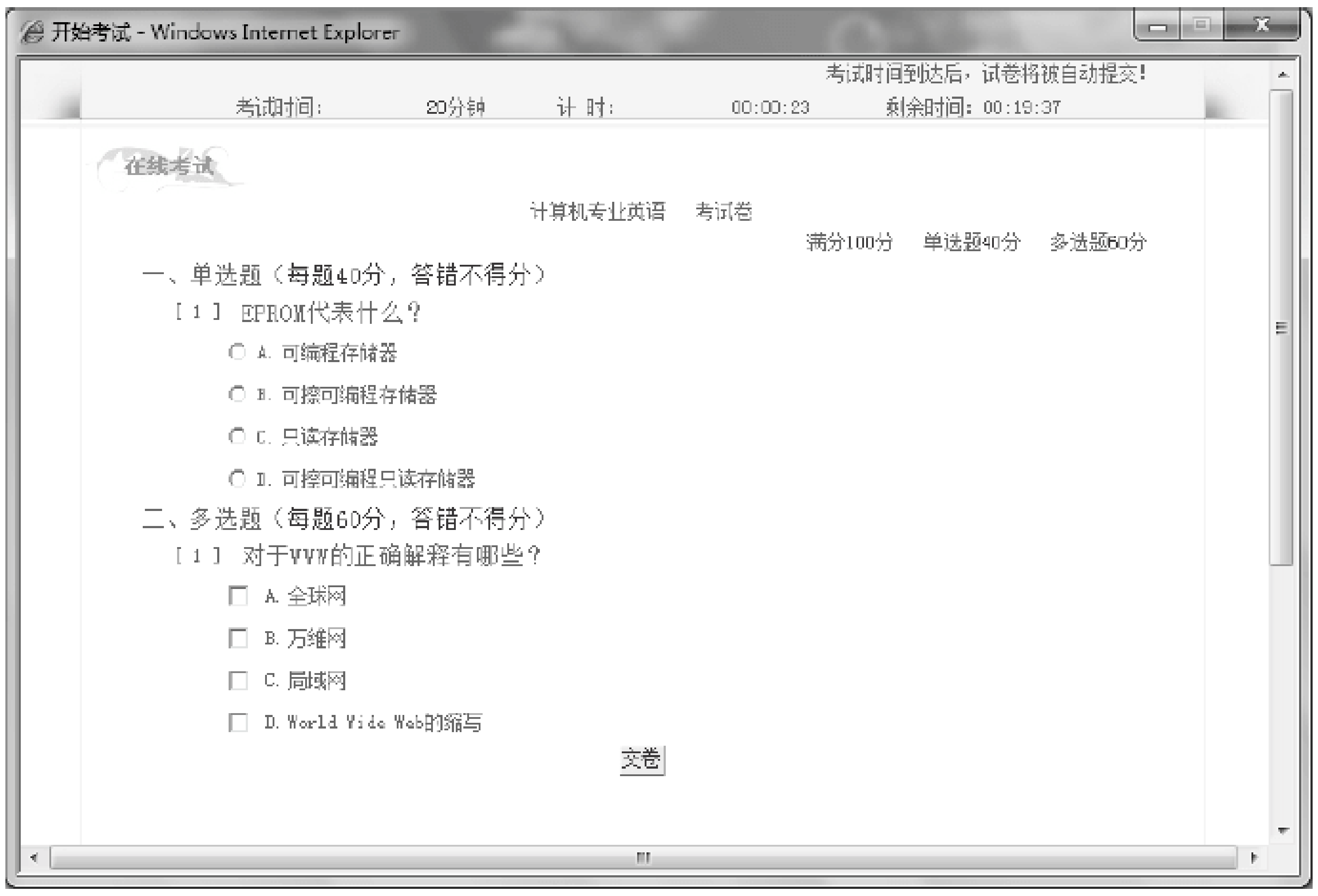


图 10.1　自动计时的开始考试页面



图 10.2　自动提交后，弹出提示对话框

关键技术

本实例主要是利用 Ajax 异步提交技术和 Servlet 技术实现的。显示在考试页面中的计时时间是在 Servlet 中设置的，需要通过 Ajax 的异步提交不断地请求 Servlet，从而获得服务器返回的最新的计时时间的数据。为了便于维护和代码的重用，可以将 Ajax 的请求方法封装到一个 JavaScript 文件中，该方法可以作为一个公共方法，在程序中使用时可以直接调用。

在 JavaScript 文件中构建 XMLHttpRequest 对象以及请求方法，代码如下：

```
/**
 * 构建 XMLHttpRequest 对象并请求服务器
 * @param reqType：请求类型（GET 或 POST）
 * @param url：服务器地址
 * @param async：是否异步请求
 * @param resFun：响应的回调函数
 * @param parameter：请求参数
 * @return：XMLHttpRequest 对象
 */
function httpRequest(reqType,url,async,resFun,parameter){
 var request = null;
```

```
if( window.XMLHttpRequest ){                                    //非 IE 浏览器，创建 XMLHttpRequest 对象
        request = new XMLHttpRequest();
}else if( window.ActiveXObject ){                               //IE 浏览器，创建 XMLHttpRequest 对象
        var arrSignatures = ["Msxml2.XMLHTTP", "Microsoft.XMLHTTP", "Microsoft.XMLHTTP", "MSXML2.XMLHTTP.5.0","MSXML2. XMLHTTP.4.0",
"MSXML2.XMLHTTP.3.0", "MSXML2.XMLHTTP"];
        for( var i = 0; i < arrSignatures.length; i++ ){
                request = new ActiveXObject( arrSignatures[i] );
                if( request || typeof( request ) == "object" )
                        break;
        }
}
if( request || typeof( request ) == "object" ){
        if(reqType.toLowerCase()=="post"){                      //以 POST 方式提交
                request.open(reqType, url, true);               //打开服务器连接
                //设置 MIME 类型
                request.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
                request.onreadystatechange = resFun;            //设置处理响应的回调函数
                parameter = encodeURI(parameter);               //将参数字符串进行编码
                request.send(parameter);                        //发送请求
        }
        else{                                                   //以 GET 方式提交
                url = url+"?"+parameter;
                request.open(reqType, url, true);               //打开服务器连接
                request.onreadystatechange = resFun;            //响应回调函数
                request.send(null);                             //发送请求
        }
}
else{
        alert( "该浏览器不支持 Ajax！ " );
}
return request;
}
```

设计过程

（1）新建 index.jsp 页面，该页面是用户访问的初始页。该页面中主要包含一个“开始考试”按钮，该按钮的 onClick 事件将调用打开考试窗口的 JavaScript 函数，关键代码如下：

```
function showWindow(){
        window.open('StartExam?action=startExam','','width=750,height=500,scrollbars=1');
}
```

（2）新建名为 StartExam 的 Servlet 实现类，该类用于创建考试的开始时间和剩余时间。在该类中，创建一个全局变量 examTime，用于记录考试时间，该变量的值是在 web.xml 中设置的，关键代码如下：

```
<servlet>
        <servlet-name>StartExam</servlet-name>
        <servlet-class>com.lh.servlet.StartExam</servlet-class>
        <init-param>
            <param-name>examTime</param-name>
            <param-value>20</param-value>
        </init-param>
</servlet>
```

在 web.xml 中设置完考试时间后，还需要在 Servlet 类的 init()方法中获得该参数值，并赋值给 Servlet 类的全局变量 examTime，关键代码如下：

```
public void init() throws ServletException {
        examTime=Integer.parseInt(getInitParameter("examTime"));  //获取配置文件中设置的考试时间
}
```

（3）在 StartExam 类中，编写用于将页面转发到开始考试页面的方法 startExam()，关键代码如下：

```
public void startExam(HttpServletRequest request,HttpServletResponse response)
        throws ServletException,IOException{
        HttpSession session = request.getSession();
        request.setAttribute("time", examTime);                 //保存考试时间
        session.setAttribute("startTime1",new Date().getTime());  //保存当前时间的毫秒数
        request.getRequestDispatcher("startExam.jsp").forward(request, response);
}
```

编写显示考试计时的方法 showStartTime()，在该方法中，首先获取保存在 Session 中的考试开始时间的毫秒值，然后获取当前时间的毫秒值，根据当前时间毫秒值和开始时间毫秒值计算出小时数、分钟数以及秒数，组合成一个新的时间字符串保存在 request 对象中。showStartTime()方法的具体代码如下：

```
public void showStartTime(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=GBK");
         HttpSession session = request.getSession();
         String startTime=session.getAttribute("startTime1").toString();
         long a=Long.parseLong(startTime);                //将开始时间转换为毫秒数
         long b=new java.util.Date().getTime();           //获取当前时间的毫秒数
         int h=(int)Math.abs((b-a)/3600000);              //获取小时
         int m=(int)(b-a)%3600000/60000;                  //获取分钟
         int s=(int)((b-a)%3600000)%60000/1000;           //获取秒数
         String hour="",minute="",second="";
         if(h<10){
                 hour= "0"+h;
         }else{
                 hour=""+h;
         }
         if(m<10){
                 minute= "0"+m;
         }else{
                 minute= ""+m;
         }
         if(s<10){
                 second = "0"+s;
         }else{
                 second = ""+s;
         }
         String time=hour+":"+minute+":"+second;          //组合已用时间
         request.setAttribute("showStartTime",time);      //将生成的时间保存到 showStartTime 参数中
        request.getRequestDispatcher("showStartTime.jsp").forward(request, response);
}
```

编写计算考试剩余时间的方法 showRemainTime()，在该方法中，首先获取步骤（3）中保存在 Session 中的考试开始时间，然后获取当前时间的毫秒值，根据当前时间毫秒值和开始时间毫秒值计算出小时数、分钟数以及秒数，组合成一个剩余时间的字符串保存在 request 对象中。showRemainTime()方法的具体代码如下：

```
public void showRemainTime(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=GBK");
         HttpSession session = request.getSession();
         String startTime=session.getAttribute("startTime").toString();
         long a=Long.parseLong(startTime);                //获取开始时间的毫秒数
         long b=new java.util.Date().getTime();           //获取当前时间的毫秒数
         long r=examTime*60000-(b-a-1000);                //计算考试剩余时间的毫秒数
         int h=(int)Math.abs(r/3600000);                  //计算小时
         int m=(int)(r)%3600000/60000;                    //计算分钟
         int s=(int)((r)%3600000)%60000/1000;             //计算秒数
         String hour="",minute="",second="";
         if(h<10){
                 hour= "0"+h;
         }else{
                 hour=""+h;
         }
         if(m<10){
                 minute= "0"+m;
         }else{
                 minute= ""+m;
         }
         if(s<10){
                 second = "0"+s;
         }else{
                 second = ""+s;
         }
         String time=hour+":"+minute+":"+second;          //组合剩余时间
```

```
        request.setAttribute("showRemainTime",time);  //将生成的时间保存到 showRemainTime 参数中
    request.getRequestDispatcher("showRemainTime.jsp").forward(request, response);          }
```

（4）新建 showStartTime.jsp 页，用于输出计时开始时间，关键代码如下：

```
<%@page contentType="text/html" pageEncoding="GBK"%>
${showStartTime}
```

（5）新建 showRemainTime.jsp 页，用于输出计时剩余时间，关键代码如下：

```
<%@page contentType="text/html" pageEncoding="GBK"%>
${showRemainTime}
```

（6）新建开始考试页面 startExam.jsp 页，在该页中通过调用 Ajax 请求方法请求 StartExam 类，获得考试的开始时间和剩余时间，关键代码如下：

```
var request1= false;
var request2 = false;
//请求 Servlet 获得开始时间
function showStartTime(){
        var url = "StartExam";
        //此处需要加&nocache="+new Date().getTime()，否则将出现时间不自动走动的情况
        var parameter="action=showStartTime&nocache="+new Date().getTime();
        request1 = httpRequest("post",url,true,callbackFunc,parameter);
}
//回调函数
function callbackFunc(){
        if( request1.readyState==4 ){
        if( request1.status == 200 ){
                        showStartTimediv.innerHTML=request1.responseText;
                }
        }
}
//请求 Servlet 获得剩余时间
function showRemainTime(){
        var url = "StartExam";
        var parameter="action=showRemainTime&nocache="+new Date().getTime();
        request2 = httpRequest("post",url,true,callbackFunc_R,parameter);
}
//回调函数
function callbackFunc_R(){
        if( request2.readyState==4 ){
        if( request2.status == 200 ){
                h=request2.responseText;
                showRemainTimediv.innerHTML=h;
                h=h.replace(/\s/g,"");        //去除字符串中的 Unicode 空白符
                        showRemainTimediv.innerHTML=h;
                        if(h=="00:00:00"){
                                form1.submit();
                        }
                }
        }
}
```

（7）为了实现页面加载后自动计时，需要在开始考试页面的<body>标签中通过 onload 事件应用 window.setInterval()方法调用 showStartTime()和 showRemainTime()函数，关键代码如下：

```
<body onLoad="showStartTime();showRemainTime();" onkeydown="keydown()">
```

秘笈心法

在构建 XMLHttpRequest 对象时，应该考虑到浏览器的兼容性。不同的浏览器下创建 XMLHttpRequest 对象的方法是不同的，支持 window.XMLHttpRequest 对象的浏览器有 Netscape 7.0+、Firefox 1.0+、Opera 8.0+、SeaMonkey 1.1+等，如果是这些浏览器时，则建立新的 XMLHttpRequest()对象。支持 window.ActiveXObject 的浏览器为 Internet Explorer 5.0+，如果是这个浏览器时，则建立 ActiveX 对象，对象名称包括 Msxml2.XMLHTTP、Microsoft.XMLHTTP、MSXML2.XMLHTTP.5.0、MSXML2.XMLHTTP.4.0、MSXML2.XMLHTTP.3.0 和 MSXML2.XMLHTTP。

实例 241　自动保存草稿

光盘位置：光盘\MR\10\241

高级

实用指数：★★★★☆

实例说明

在发表博文或编写邮件时，经常需要输入大段的文字内容，此时，如果系统出现故障或由于其他意外情况，会将已经输入的文字删除，用户就必须重新输入，既费时又费力。如果在系统中添加字段保存草稿的功能，可以有效地解决该问题。运行本实例，如图 10.3 所示，输入文章标题和内容后，每隔 5 分钟系统会自动保存一次草稿。在页面左侧的草稿箱中，将显示被保存的草稿。



图 10.3　自动保存草稿

关键技术

本实例主要利用 Ajax 技术实现无刷新操作。另外，还应用了 JavaScript 的 window 对象的 setInterval()方法实现每隔一段时间就执行一次指定的 JavaScript 函数，从而实现定时保存草稿的功能。window 对象的 setInterval()方法的语法结构如下：

```
window.setInterval(Func,Interval);
```

参数说明

❶ Func：表示将要执行的 JavaScript 代码串，也可以是指定的函数名。

❷ Interval：表示每次调用 Func 的时间间隔，以毫秒为单位。

该方法含有返回值，可以传递给 window.clearInterval()方法，从而取消对参数 Func 代码或者函数的周期执行的值。

设计过程

（1）创建一个 request.js 文件，在该文件中编写构造 XMLHttpRequest 对象并发送请求的方法。

（2）新建添加文章的页面 index.jsp，在该页的<script>标签中引用 request.js 文件，然后调用 Ajax 请求方法请求保存文章信息的页面，并编写回调函数，将响应结果输出在页面中，关键代码如下：

```
<script language="javascript" src="js/request.js"></script>
<script language="javascript">
var saveReq = false;
//保存草稿
function autoSave(){
	var title=document.getElementById("title").value;
	var content=document.getElementById("content").value;
```

```
        if(title!="") {                                    //当文章标题不为空时
                var url = "saveDraft.jsp";
                var param = "title="+title+"&content="+content;
                //调用 request.js 文件中编写的 Ajax 请求方法，并返回请求对象
                saveReq = httpRequest("post",url,true,callbackFunc_save,param);
        }
}
//回调函数
function callbackFunc_save(){
        if(saveReq.readyState == 4){
                if(saveReq.status == 200){
                        document.getElementById("sysTip").innerHTML = saveReq.responseText;
                }
        }
}
</script>
```

（3）新建保存文章信息的页面 saveDraft.jsp，在该页面中首先获取文章的请求信息，然后调用查询数据库的方法查询该草稿信息是否已被保存，如果没被保存，则调用数据库保存的方法将文章信息保存到数据库中，关键代码如下：

```
<%@ page contentType="text/html; charset=GBK" language="java" %>
<%@ page import="java.util.*" %>
<%@ page import="com.lh.dao.*" %>
<%@ page import="com.lh.bean.*"   %>
<%
        request.setCharacterEncoding("UTF-8");
        response.setCharacterEncoding("UTF-8");
        String title = request.getParameter("title");          //获取文章标题
        String content = request.getParameter("content");      //获取文章内容
        Article article = new Article();                       //创建文章类对象，并将文章信息封装到该对象中
        article.setTitle(title);                               //设置文章标题
        article.setContent(content);                           //设置文章内容
        Calendar c = Calendar.getInstance();                   //设置文章的保存时间
        String time=c.get(c.YEAR)+"-"+c.get(c.MONTH)+"-"+c.get(c.DAY_OF_MONTH)+" "+c.get(c.HOUR_OF_DAY)+":"
                        +c.get(c.MINUTE)+":"+c.get(c.SECOND);
        article.setCreateTime(time);
        boolean res = ArticleDao.getInstance().getArticle(article);
        if(!res){
                boolean b= ArticleDao.getInstance().saveArticle(article);
                if(b){
                        out.println("文章内容已经自动保存到草稿箱！ ");
                }
        }
%>
```

（4）在 index.jsp 页面中通过 window 对象的 setInterval()方法每隔 5 分钟执行一次保存草稿的操作，关键代码如下：

```
var delay=1000*60*5;                                   //定义延迟时间，这里为 5 分钟
timer=window.setInterval(autoSave, delay);             //每隔 5 分钟保存一次草稿
```

（5）编写封装文章信息的 JavaBean 类 Article、数据库连接类 DBCon 以及数据库操作的类 ArticleDao。由于篇幅有限，此处不作详细介绍，具体代码请参见本书附赠的光盘。

秘笈心法

本实例没有用 Servlet，而直接通过 JSP 页来处理请求并调用保存数据的方法。由于 JSP 页是运行在服务器端的，在运行时，JSP 页会被编译成类似于 Servlet 的类文件。所以，当处理简单的请求时可以直接使用 JSP 页来代替 Servlet。

10.2 改善用户体验

在 Ajax 应用中，可以实现在不刷新整个页面的情况下对部分数据进行更新，从而降低了网络流量，给用户

带来了更好的体验。下面将通过几个实例介绍通过 Ajax 实现改善用户体验的具体应用。

实例 242	检查用户名是否重复 光盘位置：光盘\MR\10\242	初级 实用指数：★★★★

实例说明

在实现用户注册或商品信息添加时，经常需要验证用户名或添加的商品名称是否唯一。这种验证操作应该在整个表单提交之前完成，如果通过提交整个表单来实现验证用户名是否重复，这种做法是不可取的。这样的问题应该由 Ajax 来解决，通过 Ajax 异步提交验证用户名，不仅可以有效地改善用户体验，而且提高了网页的运行速度。运行本实例，如图 10.4 所示，在用户注册表单中输入用户名“mr”，单击“[检测用户]”超链接，如果该用户名在数据库中已经存在，会弹出“用户名已存在，请重新输入！”的提示对话框。



图 10.4　检查用户名是否重复

关键技术

本实例主要是应用 Ajax 技术来实现的。Ajax 的 XMLHttpRequest 对象不仅可以通过 responseText 属性获得服务器返回的普通字符串类型的数据，而且还可以通过 responseXML 属性获得服务器返回的 XML 格式的数据。如表 10.1 所示，在 XMLHttpRequest 对象中包含了一些常用属性及说明。

表 10.1　XMLHttpRequest 对象的常用属性及说明

属　性	说　明
onreadystatechange 属性	用于指定状态改变时所触发的事件处理器。在 Ajax 中，每个状态改变时都会触发这个事件处理器，通常会调用一个 JavaScript 函数
readyState 属性	用于获取请求的状态。该属性共包括 5 个属性值。其中，0 表示未初始化，1 表示正在加载，2 表示已加载，3 表示交互中，4 表示完成
responseText 属性	用于获取服务器的响应，表示为字符串
responseXML 属性	用于获取服务器的响应，表示为 XML。这个对象可以解析为一个 DOM 对象
status 属性	用于返回服务器的 HTTP 状态码，常用的状态码有 200（表示成功）、202（表示请求被接受，但尚未成功）、400（错误的请求）、404（文件未找到）、500（内部服务器错误）

设计过程

（1）创建 request.js 文件，用于封装 Ajax 请求服务器的方法。

（2）新建用户注册表单页 index.jsp，在表单中包含一个检测用户的超链接，超链接的 onClick 事件会调用

检查用户名的 checkName()方法，关键代码如下：

```
<tr>
        <td align="center">用户名：</td>
        <td><input name="username" type="text" class="table" id="username">
         <a href="#" onClick="checkName()">[检测用户]</a> *</td>
</tr>
```

“[检测用户]”超链接调用的 checkName()方法的关键代码如下：

```
var request=false;
function checkName(){
        var name=document.getElementById("username").value;
        if(name==null||name==""){
                alert("请输入用户名！");
                document.getElementById("username").focus();
                return false;
        }
        else{
                var url= "UserServlet";                                  //服务器地址
                var param = "name="+name;                                //请求参数
              request=httpRequest("post",url,true,callbackFunc,param);   //调用 Ajax 提交请求方法
        }
}
```

（3）新建 Servlet 的实现类 UserServlet，在该类的 doPost()方法中获得通过 Ajax 异步提交的用户名，然后根据此用户名查询数据库是否存在该用户，将查询结果以 XML 格式响应给客户端，关键代码如下：

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        response.setContentType( "text/xml;charset=UTF-8" );             //设置响应正文格式为 test/xml
        //禁止缓存
        response.addHeader( "Cache-Control", "no-store,no-cache,must-revalidate" );
        response.addHeader( "Cache-Control", "post-check=0,pre-check=0" );
        response.addHeader( "Expires", "0" );
        response.addHeader( "Pragma", "no-cache" );
        PrintWriter out = response.getWriter();
        out.println( "<?xml version=\"1.0\" encoding=\"utf-8\"?>" );
        out.println( "<userName>" );
        String name = request.getParameter( "name" );                    //获取用户名
        name= java.net.URLDecoder.decode(name,"UTF-8");                  //根据 UTF-8 将字符串解码
        boolean result=UserDao.getInstance().checkUserName(name.trim());//查找数据库是否存在该用户名
        if(!result)
                out.println("<noIterance id=\"ok\"/>");
        else
                out.println("<iterance>"+name+"</iterance>");
        out.println( "</userName>" );
        out.close();
}
```

（4）在 index.jsp 页面中编写 Ajax 回调函数，获得服务器端返回的 XML 格式的信息，根据返回的 XML 信息判断用户名是否存在，并弹出提示对话框，关键代码如下：

```
//响应的回调函数
function callbackFunc(){
        if( request.readyState==4 ){
          if( request.status == 200 ){
                //获得从服务器端返回的 XML 信息，noIterance 为 XML 中的元素
                var noIterance=request.responseXML.getElementsByTagName("noIterance");
                //获得从服务器端返回的 XML 信息，iterance 为 XML 中的元素
                var iterance=request.responseXML.getElementsByTagName("iterance");
                if(typeof(noIterance)!="undefined"&& noIterance.length>0){
                        alert("恭喜您，此用户名可以注册！");
                        request=false;
                        return true;
                }
                if(typeof(iterance)!="undefined"&& iterance.length>0){
                        alert("用户名已存在，请重新输入！");
                        document.getElementById("username").focus();
                        request=false;
```

```
                    return false;
                }
            }
        }
}
```

（5）编写连接 MySQL 数据库的 DBCon 类、封装用户信息的 JavaBean 类 User 以及查询数据库中用户名是否重复的 UserDao 类。这几个类的实现比较简单，由于篇幅有限，具体代码请参见本书的附赠光盘。

秘笈心法

在通过 Servlet 回传给浏览器 XML 格式的数据时，要指定响应格式为 XML，具体代码如下：

```
response.setContentType( "text/xml;charset=UTF-8" );
```

实例 243　验证用户登录　初级

光盘位置：光盘\MR\10\243　实用指数：★★★★

实例说明

在各大网站或者 Web 企业级应用程序中，都会在首页中包含用户登录的入口。当用户输入用户名和密码并提交后，会查询数据库，验证用户名和密码是否正确。当然，验证用户登录的方法可以通过将整个页面都提交来实现，但是考虑到这种方法会影响到用户的体验，用户不想在另一个页面中发现自己登录失败，然后再后退到登录页重新登录这样多余的操作。所以应该使用 Ajax 来处理这样的问题。运行本实例，如图 10.5 所示，输入用户名和密码后，单击“登录”按钮，如果用户名和密码错误，会弹出提示对话框。



图 10.5　验证用户登录

关键技术

本实例的实现主要是通过 Ajax 技术将用户名和密码信息异步提交到 Servlet 中，然后在 Servlet 中根据获得的用户名和密码查询数据库，最后响应给客户端 XML 格式的查询结果。客户端通过 Ajax 回调函数来获得服务器返回的 XML 信息，并根据 XML 信息给出相应的提示信息。

设计过程

（1）创建 request.js 文件，在该文件中编写 Ajax 请求方法。

（2）新建用户登录页 index.jsp，在该页中首先导入 request.js 文件，然后在<script>标签中编写 Ajax 请求服务器的 JavaScript 函数和验证用户名和密码是否合法的 JavaScript 函数，关键代码如下：

```
<script type="text/javascript" src="js/request.js"></script>
<script type="text/javascript">
var opinionRequest=false;
//登录请求
function enter(){
        if(checkInsert()){
                var name=document.getElementById("username").value;              //用户名
                var pwd = document.getElementById("userpwd").value;              //密码
                var url="UserLoginServlet";                                      //服务器地址
                var param ="action=checkLogin&name="+name+"&pwd="+pwd;           //请求参数
```

```
            opinionRequest=httpRequest("post",url,true,callbackFunc,param);    //调用请求方法

        }
}
//验证用户名密码是否合法
function checkInsert(){
        var name = document.getElementById("username").value;
        var pwd = document.getElementById("userpwd").value;
        if(name==null||name==""){
                alert("请输入您的账号！ ");
                document.getElementById("username").focus();
                return false;
        }
        if(pwd==null||pwd==""){
                alert("请输入您的密码！ ");
                document.getElementById("userpwd").focus();
                return false;
        }
        if(pwd!=null&&pwd!=""){
                if(pwd.length<6||pwd.length>16){
                        alert("输入有误，密码长度为 6～16 位！ ");
                        document.getElementById("userpwd").focus();
                        return false;
                }
        }
        return true;
}
</script>
```

（3）创建 Servlet 的实现类 UserLoginServlet，该类主要用于验证用户登录。在 doPost()方法中获得用户名和密码信息，然后根据用户名和密码查询数据库，最后响应给客户端 XML 格式的结果，关键代码如下：

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        response.setCharacterEncoding("UTF-8");
        response.setContentType( "text/xml;charset=UTF-8" );                //设置响应格式为 test/xml
        //禁止缓存
        response.addHeader( "Cache-Control", "no-store,no-cache,must-revalidate" );
        response.addHeader( "Cache-Control", "post-check=0,pre-check=0" );
        response.addHeader( "Expires", "0" );
        response.addHeader( "Pragma", "no-cache" );
        PrintWriter out = response.getWriter();
        out.println( "<?xml version=\"1.0\" encoding=\"utf-8\"?>" );
        out.println( "<checkLogin>" );
        String name = request.getParameter( "name" );                     //获取用户名
        String pwd = request.getParameter("pwd");                         //获取密码
        name= java.net.URLDecoder.decode(name,"UTF-8");                   //根据 UTF-8 将字符串解码
        User user = new User();
        user.setName(name);
        user.setPwd(pwd);
        boolean result=UserDao.getInstance().checkLogin(user);
        if(result==true)
                out.println("<hasUser id=\"ok\"/>");
        else
                out.println("<noUser>"+name+"</noUser>");
        out.println( "</checkLogin>" );
        out.close();
}
```

（4）在 web.xml 文件中配置 UserLoginServlet 类，关键代码如下：

```
<servlet>
   <description>This is the description of my J2EE component</description>
   <display-name>This is the display name of my J2EE component</display-name>
   <servlet-name>UserLoginServlet</servlet-name>
   <servlet-class>com.lh.servlet.UserLoginServlet</servlet-class>
</servlet>
<servlet-mapping>
   <servlet-name>UserLoginServlet</servlet-name>
   <url-pattern>/UserLoginServlet</url-pattern>
</servlet-mapping>
```

```
<welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
```

（5）编写连接 MySQL 数据库的 DBCon 类、封装用户信息的 JavaBean 类 User 以及数据库操作的类 User Dao。这几个类的实现比较简单，由于篇幅有限，具体代码请参见本书的附赠光盘。

秘笈心法

应用 Ajax 请求时，为了避免出现中文乱码问题，已经通过 JavaScript 中的 encodeURI()方法对请求数据进行了编码。所以，在 Servlet 中获得这些请求数据时，应该通过 java.net.URLDecoder 类中的 decode()方法对请求数据进行解码，语法格式如下：

```
name= java.net.URLDecoder.decode(name,"UTF-8");   //根据 UTF-8 将字符串解码
```

实例 244　限时竞拍

光盘位置：光盘\MR\10\244

高级

实用指数：★★★★

实例说明

在网上拍卖的网站中，主要功能就是限时竞拍。本实例将介绍如何通过 Ajax 实现限时竞拍。运行本实例，首先进入的是拍品列表页面，默认显示的是“正在进行的拍卖”的拍品列表，选择“已经结束的拍卖”选项卡，将显示已经结束拍卖的拍品列表，在该页面中单击某一拍品列表右侧的“查看明细”按钮，可以看到该拍品的基本信息及出价记录。在正在进行拍卖的拍品列表页面中，单击某一拍品列表右侧的“我要出价”按钮，将进入拍品出价页面，在该页面中将显示拍品的基本信息和当前出价情况，如果要为该拍品出价，需要先登录网站，登录后将显示如图 10.6 所示的页面，这时就可以为拍品出价了。



图 10.6　限时竞拍

关键技术

在实现限时竞拍时，最重要的技术就是应用 Ajax 实现异步操作，主要是应用 Ajax 实时获取商品拍卖的剩余时间以及实时获取商品的最高出价信息。

设计过程

（1）创建 request.js 文件，在该文件中编写 Ajax 请求方法。

（2）创建限时竞拍的首页 index.jsp，该页面为一个中转页，用于将页面重定向到拍品列表页面，关键代码如下：

```
<%response.sendRedirect("AuctionServlet?action=getRes&state=0"); %>
```

（3）创建 Servlet 的实现类 AuctionServlet，并且在该 Servlet 的 doPost()方法中获取 action 参数的值，根据 action 参数值的不同来调用不同的方法，关键代码如下：

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException{
    String action = request.getParameter("action");
    if ("getRes".equals(action))
        getRes(request, response);                                  //查询拍品信息的方法
    else if ("getSingleRes".equals(action))
        getSingleRes(request, response);                            //查询单个拍品信息
    else if ("getRemainTime".equals(action))
        getRemainTime(request, response);                           //实时获取剩余时间
    else if ("getBidInfo".equals(action))
        getBidInfo(request, response);                              //获取当前拍品的最高出价信息
    else if ("logout".equals(action))
        logout(request, response);                                  //退出
    else if ("login".equals(action))
        login(request, response);                                   //登录
    else if ("bidding".equals(action))
        bid(request, response);                                     //保存当前出价信息
}
```

在 AuctionServlet 中编写 getRes()方法，获取请求中的拍品信息，然后查询数据库，从数据库中获取相应的拍品信息，最后将页面转发到拍品列表页面。getRes()方法的具体代码如下：

```
private void getRes(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
response.setCharacterEncoding("UTF-8");
int isEnd = Integer.parseInt(request.getParameter("state"));                //获取状态参数
//查询拍品信息
ConnDB conn = new ConnDB();
String sql = "SELECT r.*,b.heightPrice FROM tb_res r LEFT JOIN (select resId ,max(bid) heightPrice FROM tb_biddinglist GROUP BY resId) b ON
r.id=b.resId WHERE isEnd="+ isEnd;
ResultSet rs = conn.executeQuery(sql);
List<ResForm> list = new ArrayList<ResForm>();                             //创建保存拍品信息的 List 集合对象
try {
        while (rs.next()) {
                ResForm res = new ResForm();
                res.setId(rs.getInt(1));                                    //ID 号
                res.setName(rs.getString(2));                               //拍品名称
                res.setStartPrice(rs.getFloat(3));                          //起始价
                res.setBreadth(rs.getInt(4));                               //加价幅度
                res.setStartTime(java.text.DateFormat.getDateTimeInstance().parse(rs.getString(5)));   //起始时间
                res.setEndTime(java.text.DateFormat.getDateTimeInstance().parse(rs.getString(6)));     //结束时间
                res.setHit(rs.getInt(7));                                   //浏览次数
                res.setIsEnd(rs.getInt(8));                                 //是否结束
                res.setPicture("images/goods/" + rs.getString(9));
                res.setHeightPrice(rs.getString(10) == null ? 0 : Float.valueOf(rs.getString(10)));  //设置最高竞价
                list.add(res);
        }
} catch (Exception e) {
        e.printStackTrace();
}
conn.close();
request.setAttribute("resList", list);                                     //将查询到的拍品信息保存到 request 中
String state = isEnd == 0 ? "正在进行的拍卖" : "已经结束的拍卖";
request.setAttribute("state", state);                                      //保存当前状态到 request 中
request.getRequestDispatcher("main.jsp").forward(request, response);
}
```

（4）创建商品拍卖列表页 main.jsp，关键代码如下：

```
<!--根据状态显示用于控制显示内容的选项卡-->
<div style="height:34px; width:97%; border-bottom:solid 5px #BADF75;">
        <c:if test="${requestScope.state=='正在进行的拍卖'}">
```

```
                <img src="images/time2_03.jpg" width="120" height="34">
                <a href="AuctionServlet?action=getRes&state=1">
                    <img src="images/time2_04.jpg" width="121" height="34" border="0">
                </a>
        </c:if>
        <c:if test="${requestScope.state!='正在进行的拍卖'}">
                <a href="AuctionServlet?action=getRes&state=0">
                    <img src="images/time4_03.jpg" width="120" height="34" border="0">
                </a>
                <img src="images/time4_04.jpg" width="121" height="34" border="0">
        </c:if>
</div>
<!--显示拍品列表-->
<div id="goodsList">
  <table width="97%" height="59" border="0" cellpadding="0" cellspacing="0">
  <c:forEach items="${requestScope.resList}" var="res">
          <tr>
            <td align="center">
                    <img src="${res.picture} " width="80" height="66">  
              </td>
            <td>${res.name}<br>浏览次数：${res.hit}</td>
            <td >起拍价：${res.startPrice}（元）<br>
                  最高竞价：<font color="red">${res.heightPrice}</font>（元）
               </td>
          <c:if test="${requestScope.state=='正在进行的拍卖'}">
            <td align="center">
                <a href="AuctionServlet?action=getSingleRes&id=${res.id} ">
                <img src="images/bid.jpg" border="0"></a>
              </td>
            </c:if>
          <c:if test="${requestScope.state!='正在进行的拍卖'}">
            <td align="center">
                    <a href="AuctionServlet?action=getSingleRes&id=${res.id} ">
                    <img src="images/detail.jpg" border="0"></a>
              </td>
        </c:if>
          </tr>
</c:forEach>
</table>
</div>
```

（5）在 AuctionServlet 中编写 getSingleRes()方法，该方法用于查询单个拍品信息。在该方法中，首先获取要查询的拍品的 ID 号，并从数据库中查询指定拍品的详细信息，然后获取该拍品的详细出价记录，再将浏览次数加 1，最后将拍品信息、出价记录和拍品 ID 保存到 request 中，并将页面转发到显示拍品详细信息的页面，关键代码如下：

```
private void getSingleRes(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
response.setCharacterEncoding("UTF-8");
int id = Integer.parseInt(request.getParameter("id"));          //获取要查询拍品的 ID 号
//查询拍品信息
ConnDB conn = new ConnDB();
String sql = "select t1.*,t2.username from (SELECT r.*,b.heightPrice,b.bidCount FROM tb_res r LEFT JOIN (select resId ,max(bid) heightPrice,count(*) bidCount FROM tb_biddinglist GROUP BY resId) b ON r.id=b.resId) t1 INNER JOIN (select r.id,u.username from tb_res r LEFT JOIN (select b.resId,u.userName from (select b.* from tb_biddinglist b inner join (select resId,max(bid) maxbid from tb_biddinglist group by resId) m ON b.resId=m.resId AND b.bid=m.maxbid) b INNER JOIN tb_user u ON b.bidder=u.id )u ON r.id=u.resId) t2 ON t1.id=t2.id WHERE t1.id="+ id + "";
ResultSet rs = conn.executeQuery(sql);                          //执行查询语句
ResForm res = new ResForm();
try {
        if (rs.next()) {
                res.setId(rs.getInt(1));
                res.setName(rs.getString(2));
                res.setStartPrice(rs.getFloat(3));
                res.setBreadth(rs.getInt(4));
                Date startT = java.text.DateFormat.getDateTimeInstance().parse(rs.getString(5))
                Date endT = java.text.DateFormat.getDateTimeInstance().parse(rs.getString(6))
                res.setStartTime(startT);
                res.setEndTime(endT);
```

```
                res.setHit(rs.getInt(7));
                res.setIsEnd(rs.getInt(8));
                res.setPicture("images/goods/" + rs.getString(9));
                float heightPrice = (rs.getString(10) == null)?0.0F
                                                    :Float.valueOf(rs.getString(10)).floatValue();
                res.setHeightPrice(heightPrice);                         //设置最高竞价
                int bidCount = (rs.getString(11) == null) ? 0 : Integer.parseInt(rs.getString(11));
                res.setBidCount(bidCount);
                res.setBidder(rs.getString(12) == null ? "暂无" : rs.getString(12));
        }
} catch (Exception e) {
        e.printStackTrace();
}
/*********** 获取出价记录 *****************************/
String sql_list = "SELECT b.*,u.userName FROM tb_biddinglist b INNER JOIN tb_user u ON b.bidder=u.id WHERE b.resId="+ id + " ORDER BY
b.bid DESC";
ResultSet rs_list = conn.executeQuery(sql_list);
List<BidListForm> list_bidlist = new ArrayList<BidListForm>();
try {
        int flag = 0;
        while (rs_list.next()) {
                BidListForm bidlist = new BidListForm();
                //将最后出价的记录标记为领先状态，其他记录标记为出局状态
                if (flag == 0) {
                        bidlist.setState("<font style='color:red'>领先</font>");
                        flag = 1;
                } else {
                        bidlist.setState("<font style='color:green'>出局</font>");
                }
                bidlist.setId(rs_list.getInt(1));                        //获取记录 ID
                bidlist.setBid(rs_list.getFloat(3));                     //获取出价
                bidlist.setBidTime(java.text.DateFormat.getDateTimeInstance()
                                .parse(rs_list.getString(5)));           //获取出价时间
                bidlist.setBidder(rs_list.getString(6));                 //获取出价人 ID
                list_bidlist.add(bidlist);
        }
} catch (Exception e) {
        e.printStackTrace();
}
/***********将浏览次数加 1****************************/
String sql_updateHit = "UPDATE tb_res SET hit=hit+1 WHERE id=" + id  + "";
conn.executeUpdate(sql_updateHit);                                       //执行更新语句
/****************************************************/
conn.close();                                                            //关闭数据库连接
request.setAttribute("resInfo", res);                                    //将查询到的拍品信息保存到 request 中
request.setAttribute("id", id);
request.setAttribute("bidInfo", list_bidlist);                           //将出价记录保存到 request 中
request.getRequestDispatcher("detail.jsp").forward(request, response);
}
```

（6）在 detail.jsp 页面中，编写创建 Ajax 对象的方法 getRemainTime()，用于向服务器发送请求，实时获取剩余时间，关键代码如下：

```
<script src="js/request.js"></script>
<script language="javascript">
var timeRequest = false;
var bidRequest =false;
var listRequest = false;
/**************实时获取剩余时间***********************************************/
function getRemainTime() {
var url="AuctionServlet";                                                //服务器地址
//需要加&nocache="+new Date().getTime()，否则将出现剩余时间不更新的情况
var param ="action=getRemainTime&id=${requestScope.id}&nocache="+ new Date().getTime();
timeRequest=httpRequest("post",url,true,callbackFunc_time,param);        //调用请求方法
}
```

（7）在 AuctionServlet 中再编写 getRemainTime()方法，用于获取剩余时间，具体代码如下：

```
private void getRemainTime(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
```

```
response.setCharacterEncoding("UTF-8");                    //设置响应的编码方式
int id = Integer.parseInt(request.getParameter("id"));     //获取拍品的 ID 号
//查询拍品信息
ConnDB conn = new ConnDB();
String sql = "SELECT *FROM tb_res WHERE id=" + id + "";
ResultSet rs = conn.executeQuery(sql);
Date endTime = null;                                       //结束时间
try {
        if (rs.next()) {
                endTime = java.text.DateFormat.getDateTimeInstance().parse(
                                rs.getString(6));          //获取结束时间
        }
} catch (Exception e) {
        e.printStackTrace();
}
/************ 计算剩余时间 ***************************/
Date nowTime = new Date();
long remainTime = (endTime.getTime() - nowTime.getTime());
long remainDay = remainTime / (24 * 60 * 60 * 1000);       //计算剩余的天数
remainTime = remainTime - remainDay * (24 * 60 * 60 * 1000);
long remainHour = remainTime / (60 * 60 * 1000);           //计算剩余的小时数
remainTime = remainTime - remainHour * (60 * 60 * 1000);
long remainMinute = remainTime / (60 * 1000);              //计算剩余的分钟数
remainTime = remainTime - remainMinute * (60 * 1000);
long remainSecond = remainTime / 1000;                     //计算剩余的秒数
/*****************************************/
response.setContentType("text/html");                      //设置应答类型为 html
PrintWriter out = response.getWriter();
if (remainDay > 0 || remainHour > 0 || remainMinute > 0      || remainSecond > 0) {
        out.println("<font    style='color:red'>" + remainDay
                        + "</font>天<font    style='color:red'>" + remainHour
                        + "</font>时<font style='color: red'>" + remainMinute
                        + "</font>分<font style='color: red'>" + remainSecond
                        + "</font>秒");
} else {
        out.println("end");
        //更新拍品状态
        String sql_updateState = "UPDATE tb_res SET isEnd=1 WHERE id=" + id+ "";
        conn.executeUpdate(sql_updateState);               //执行更新语句
}
/*****************************************/
conn.close();
}
```

（8）在 detail.jsp 页面中，编写 Ajax 回调函数获取最新的商品拍卖剩余时间，具体代码如下：

```
function callbackFunc_time(){
if( timeRequest.readyState==4 ){                           //判断响应是否完成
        if( timeRequest.status == 200 ){                   //判断响应是否成功
                var remain = timeRequest.responseText;     //获取剩余时间
                if ( remain.replace(/\s/g, "").indexOf("end") != -1) {
                        window.clearInterval(t);
                        document.getElementById("ctrlBtn").innerHTML = "<img src='images/end.jpg' border='0'>";
                        document.getElementById("remain").innerHTML = "<font style='color:green;'>已经结束</font>";
                        document.getElementById("state").innerHTML="已经结束的拍卖";
                        document.getElementById("bidArea").style.display = "none";
                } else {
                        document.getElementById("remain").innerHTML = remain;
                }
        }
}
}
```

（9）实现限时竞拍功能，还需要应用 Ajax 实现实时获取最高出价信息。由于实现过程与实时获取最新的商品拍卖剩余时间类似，所以此处不再给出具体代码。具体代码可以参考本书的附赠光盘。

（10）在 detail.jsp 页中，想要实时获取商品拍卖的最新剩余时间和最高出价信息，需要在页面加载时通过 window 对象的 setInterval()方法每隔一秒就调用 Ajax 请求方法请求一次服务器，关键代码如下：

```
//页面载入后要执行的操作
window.onload = function() {
        getRemainTime();
        t=window.setInterval("getRemainTime()", 1000);        //每隔 1 秒钟调用一次 getRemainTime()方法
        getBidInfo();
        window.setInterval("getBidInfo()", 60 * 1000);        //每隔 1 分钟获取一次出价信息
}
```

（11）实现用户登录，并且在用户登录后提供为拍品出价功能。由于用户登录比较简单，这里不作详细介绍。

秘笈心法

微软 Internet Explorer 在缓存暂存文件夹中存放获取 GET 的响应，由于缓存不会一直更新内容，所以使用频率高时，可能会造成回传数据都相同，此时可以考虑改用 POST 的方式，不通过缓存获取回传数据。

通常，为解决 GET 的缓存问题，在 Ajax 请求指定 URL 的参数时会加上一个变量，保证每次请求的 URL 地址都不同，例如：

```
xmlHttp.open("GET","AuctionServlet?nocache="+new Date().getTime());
```

由于每次获得的当前时间的值都不同，这样每次请求的 URL 地址都会识别为不同的链接，从而获得最新的回传数据。

实例 245 带进度条的文件上传

光盘位置：光盘\MR\10\245

高级

实用指数：★★★★☆

实例说明

在实际的 Web 应用开发或网站开发过程中，经常需要实现文件上传的功能。在文件上传过程中，经常需要用户进行长时间的等待，为了让用户及时了解上传进度，可以在上传文件的同时显示文件的上传进度条。本实例即可实现这一功能。运行本实例，如图 10.7 所示，访问文件上传页面，单击“浏览”按钮选择要上传的文件，注意文件不能超过 50MB，否则系统将给出错误提示。选择完要上传的文件后，单击“提交”按钮，将会上传文件并显示上传进度。



图 10.7　上传文件并显示进度条

关键技术

本实例主要是应用开源的 Common-FileUpload 组件来实现分段文件上传，从而实现在上传过程中，不断获取上传进度。下面对 Common-FileUpload 组件进行详细介绍。

Common-FileUpload 组件是 Apache 组织下的 jakarta-commons 项目下的一个子项目，使用该组件可以方便地将 multipart/form-data 类型请求中的各种表单域解析出来，它需要另一个名为 Common-IO 的组件的支持。这两个组件包文件可以在 http://commons.apache.org 网站上下载得到。

1. 创建上传对象

在应用 Common-FileUpload 组件实现文件上传时，需要创建一个工厂对象，并根据该工厂对象创建一个新的文件上传对象，具体代码如下：

```
DiskFileItemFactory factory = new DiskFileItemFactory();
ServletFileUpload upload = new ServletFileUpload(factory);
```

2. 解析上传请求

创建一个文件上传对象后，就可以应用该对象来解析上传请求，获取全部的表单项，可以通过文件上传对

象的 parseRequest()方法来实现。parseRequest()方法的语法结构如下：

```
public List parseRequest(HttpServletRequest request) throws FileUploadException
```

3. FileItem 类

在 Common-FileUpload 组件中，无论是文件域还是普通表单域，都应当成 FileItem 对象来处理。如果该对象的 isFormField()方法返回值为 true，则表示是一个普通表单域，否则为一个文件域。在实现文件上传时，可以通过 FileItem 类的 getName()方法获得上传文件的文件名，通过 getSize()方法获得上传文件的大小。

设计过程

（1）创建 request.js 文件，在该文件中编写 Ajax 请求方法。

（2）新建文件上传页 index.jsp，在该页中添加用于获得上传文件信息的表单以及表单元素，并添加用于显示进度条的<div>标签和显示百分比的<span>标签，关键代码如下：

```
<form enctype="multipart/form-data" method="post" action="UpLoad?action=uploadFile">
        请选择上传的文件：<input name="file" type="file" size="34">
          //文件大小请控制在 50MB 以内
        <div id="progressBar" class="prog_border" align="left">
          <img src="images/progressBar.gif" width="0" height="13" id="imgProgress"></div>
        <span id="progressPercent" style="width:40px;display:none">0%</span>
        <input name="Submit" type="button" value="提交" onClick="deal(this.form)">
          <input name="Reset" type="reset" class="btn_grey" value="重置"></td>
</form>
```

（3）新建上传文件的 Servlet 实现类 UpLoad。在该类中编写实现文件上传的方法 uploadFile()，在该方法中通过 Common-FileUpload 组件实现分段上传文件，并计算上传百分比，实时保存到 Session 中，关键代码如下：

```
public void uploadFile(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=GBK");
    request.setCharacterEncoding("GBK");
    HttpSession session=request.getSession();
    session.setAttribute("progressBar",0);                          //定义指定上传进度的 Session 变量
    String error = "";
    int maxSize=50*1024*1024;                                       //单个上传文件大小的上限
    DiskFileItemFactory factory = new DiskFileItemFactory();        //创建工厂对象
    ServletFileUpload upload = new ServletFileUpload(factory);      //创建一个新的文件上传对象
    try {
        List items = upload.parseRequest(request);                  //解析上传请求
        Iterator itr = items.iterator();                            //枚举方法
        while (itr.hasNext()) {
            FileItem item = (FileItem) itr.next();                  //获取 FileItem 对象
            if (!item.isFormField()) {                              //判断是否为文件域
                if (item.getName() != null && !item.getName().equals("")) {  //是否选择了文件
                    long upFileSize=item.getSize();                 //上传文件的大小
                    String fileName=item.getName();                 //获取文件名
                    if(upFileSize>maxSize){
                        error="您上传的文件太大，请选择不超过 50MB 的文件";
                        break;
                    }
                    //此时文件暂存在服务器的内存中
                    File tempFile = new File(fileName);             //构造文件目录临时对象
                    String uploadPath = this.getServletContext().getRealPath("/upload");
                    File file = new File(uploadPath,tempFile.getName());
                    InputStream is=item.getInputStream();
                    int buffer=1024;                                //定义缓冲区的大小
                    int length=0;
                    byte[] b=new byte[buffer];
                    double percent=0;
                    FileOutputStream fos=new FileOutputStream(file);
                    while((length=is.read(b))!=-1){
                        percent+=length/(double)upFileSize*100D;    //计算上传文件的百分比
                         fos.write(b,0,length);                     //向文件输出流写读取的数据
```

```
                        session.setAttribute("progressBar",Math.round(percent));
                    }
                    fos.close();
                        Thread.sleep(1000);                         //线程休眠 1 秒
                } else {
                    error="没有选择上传文件！";
                }
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
        error = "上传文件出现错误：" + e.getMessage();
    }
    if (!"".equals(error)) {
        request.setAttribute("error", error);
        request.getRequestDispatcher("error.jsp").forward(request, response);
    }else {
        request.setAttribute("result", "文件上传成功！");
        request.getRequestDispatcher("upFile_deal.jsp").forward(request, response);
    }
}
```

（4）在文件上传页 index.jsp 中，导入编写的 Ajax 请求方法的 request.js 文件，并编写获取上传进度的 Ajax 请求方法和 Ajax 回调函数，关键代码如下：

```
<script language="javascript" src="js/request.js"></script>
<script language="javascript">
var request = false;
function getProgress(){
        var url="showProgress.jsp";                              //服务器地址
        var param ="nocache="+new Date().getTime();              //每次请求 URL 参数都不同，避免上传时进度条不动
        request=httpRequest("post",url,true,callbackFunc,param); //调用请求方法
}
//Ajax 回调函数
function callbackFunc(){
 if( request.readyState==4 ){                                     //判断响应是否完成
          if( request.status == 200 ){                            //判断响应是否成功
                var h = request.responseText;                     //获得返回的响应数据
                h=h.replace(/\s/g,"");                            //去除字符串中的 Unicode 空白符
                document.getElementById("progressPercent").style.display="";    //显示百分比
                progressPercent.innerHTML=h+"%";                  //显示完成的百分比
                document.getElementById("progressBar").style.display="block";   //显示进度条
                document.getElementById("imgProgress").width=h*(235/100);        //显示完成的进度
          }
 }
}
</script>
```

（5）编写 showProgress.jsp 页面，在该页面中应用 EL 表达式输出保存在 session 域中的上传进度条的值，具体代码如下：

```
<%@page contentType="text/html" pageEncoding="GBK"%>
${progressBar}
```

（6）编写表单提交按钮 onClick 事件所调用的 JavaScript 方法，在该方法中通过 window 对象的 setInterval()方法每隔一定时间请求一次服务器，获得最新的上传进度，关键代码如下：

```
function deal(form){
        form.submit();                                          //提交表单
        timer=window.setInterval("getProgress()",500);          //每隔 500 毫秒获取一次上传进度
}
```

秘笈心法

当通过 POST 方式提交请求数据时，必须通过 XMLHttpRequest 对象的 setRequestHeader()方法设置 MIME 类型，否则服务器无法正确解析传送过来的参数，代码如下：

```
request.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
```

实例 246　仿 Google Suggest 自动完成　高级

光盘位置：光盘\MR\10\246　实用指数：★★★★

实例说明

Google Suggest 也就是 Google 搜索引擎，含义就是当用户输入一个或几个关键字后，会自动列出部分最常用的搜索词汇，帮助用户节省输入时间并可以检查拼写错误。现在这种搜索技术被广泛应用在 Web 应用中。本实例将实现一个类似 Google Suggest 的自动完成功能。运行本实例，在搜索文本框中输入“java”后，在下面将显示以 java 开头的关键字列表，如图 10.8 所示。用户可以通过↑或↓方向键选择所需的关键字，当选中某一个关键字后，按下 Enter 键，关键字列表将关闭。如果用户输入的关键字在系统的数据库中不存在，单击“搜索结果”按钮后，该关键字将自动添加到系统的数据库中。



图 10.8　仿 Google Suggest 自动完成

关键技术

本实例实现的关键是当用户在文本框中输入数据时，需要使用文本框的 onpropertychange 事件，当文本框的值改变时就会触发该事件。因此可以应用该事件来调用 Ajax 请求方法，实时请求服务器，查询数据库指定表中是否包含相匹配的关键字，并将这些查询出的匹配数据响应给客户端。客户端通过 Ajax 回调函数获得这些匹配的关键字数据，再通过 JavaScript 函数将这些数据显示在文本框下方。

设计过程

（1）创建 request.js 文件，用于封装 Ajax 请求服务器的方法。

（2）新建 index.jsp 页面，在该页中导入 request.js 文件，然后编写一个 JavaScript 方法，在该方法中调用 request.js 文件编写的 Ajax 请求方法，用于向服务器发送请求。在用户按下除 Enter 键、↑和↓方向键以外的按键时，获取符合条件的关键字，关键代码如下：

```
<script src="js/request.js"></script>
<script language="javascript">
var request = false;
var size=0;                                         //总条目
var curSize=-1;                                     //当前条目
//Ajax 请求方法
function getSuggest() {
 if(event.keyCode!=13 && event.keyCode!=38 && event.keyCode!=40){
        curSize=-1;                                 //当前条目
        var suggest = document.getElementById("suggest").value;
        var url="KeyServlet";                       //服务器地址
        //注意关键字 suggest 的值要用 encodeURL 编码，否则可能出现乱码
        var param ="action=getKey&inputKey="+encodeURI(suggest)+"&nocache=" + new Date().getTime();
        request=httpRequest("post",url,true,callbackFunc,param);      //调用请求方法
 }
}
```

（3）创建 Servlet 的实现类 KeyServlet，用于处理 Ajax 请求，在该类的 doGet()方法中获得 action 参数值，

如果 action 参数值等于 getKey，则调用 getKey()方法从数据库中获取符合条件的关键字。doGet()方法的关键代码如下：

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    String action = request.getParameter("action");                //获取 action 参数的值
    if ("getKey".equals(action)) {
        getKey(request, response);                                 //查询符合条件的关键字
    }
}
```

（4）在 KeyServlet 中编写 getKey()方法，在该方法中首先获取输入的查询关键字，并且设置响应的编码方式及响应类型，然后创建 XML 文档对象以及根节点，再从数据表中查询出符合条件的关键字，并设置为 XML 文档对象的根节点的子节点，最后将 XML 文档输出到浏览器。getKey()方法的具体代码如下：

```
private void getKey(HttpServletRequest request,HttpServletResponse response)
    throws ServletException, IOException {
    request.setCharacterEncoding("UTF-8");
    response.setCharacterEncoding("UTF-8");
    response.setContentType("application/xml");                    //设置应答类型为 XML
    String inputKey=request.getParameter("inputKey");              //获取输入的关键字
    if(!inputKey.equals("")&&inputKey!=null){
        Document document = DocumentHelper.createDocument();       //创建 XML 文档对象
        Element keys = DocumentHelper.createElement("keys");       //创建普通节点
        document.setRootElement(keys);                             //将 keys 设置为根节点
        //查询符合条件的关键字
        ConnDB conn = new ConnDB();
        String sql = "SELECT * FROM tb_key WHERE keyWord LIKE '"+inputKey+
                "%' AND keyWord !='"+inputKey+"' ORDER BY keyWord ASC";
        ResultSet rs = conn.executeQuery(sql);                     //执行查询语句
        try {
            while (rs.next()) {
                Element key = keys.addElement("keyWord");          //添加 keyWord 节点
                String keyWord = rs.getString(2);
                key.setText(keyWord);                              //设置 keyWord 节点的内容为关键字
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        conn.close();                                              //关闭数据库连接
        OutputFormat format = new OutputFormat();                  //创建 OutputFormat 对象
        format.setEncoding("UTF-8");                               //设置写入流编码
        PrintWriter out = response.getWriter();                    //获取 JSP 的内置对象 out
        XMLWriter writer = new XMLWriter(out, format);             //实例化 XMLWriter 对象
        writer.write(document);                                    //向流中写入数据
        writer.close();                                            //关闭 XMLWriter 对象
    }
}
```

（5）在 index.jsp 文件中编写 Ajax 的回调函数。在该函数中，首先获取 XML 格式的关键字，以及文档中 id 为 content 的对象，然后设置总条目数，并且清空提示内容对象，最后获取 XML 文档中的 keyWord 节点，也就是关键字列表，如果关键字个数大于 0，则通过 for 循环调用 addContent()方法将这些关键字添加到 content 对象中，并且显示提示框，否则隐藏提示框，关键代码如下：

```
//Ajax 回调函数
function callbackFunc(){
if( request.readyState==4 ){                                       //判断响应是否完成
    if( request.status == 200 ){                                   //判断响应是否成功
        var doc = request.responseXML;                             //获取 XML 格式的关键字
        var contentObj = document.getElementById("content");       //获取 content 对象
        size=contentObj.childNodes.length;                         //设置条目数
        //清空提示内容对象
        while(true){
            if(contentObj.hasChildNodes()){
                contentObj.removeChild(contentObj.firstChild);     //移除指定的节点
            }else{
                break;                                             //跳出 while 循环
```

```
                        }
                    }
                    var content = doc.getElementsByTagName("keyWord");              //获取 XML 文档中的 keyWord 节点
                    if(content.length>0){
                        for(var i=0;i<content.length;i++){
                            addContent(content(i).firstChild.nodeValue);
                        }
                        document.getElementById("toolBox").style.visibility="visible";    //显示提示框
                    }else{
                        document.getElementById("toolBox").style.visibility="hidden";     //隐藏提示框
                    }
                }
}
}
```

（6）编写 addContent()方法用于为 content 对象添加一个子对象。addContent()方法的关键代码如下：

```
function addContent(content){
    var contentObj = document.getElementById("content");                    //获取 content 对象
    var divContent = document.createElement("div");                         //创建一个 DIV 对象
    divContent.innerHTML=content;                                           //设置 div 标记的内容
    divContent.style.paddingLeft="2px";                                     //设置文字与左边框的距离
    divContent.onmouseover = function(){                                    //添加鼠标移入事件
        this.style.background="#CE83FF";                                    //当鼠标在关键词上时改变背景色
        this.style.color="#FFFFFF";                                         //当鼠标在关键词上时改变文字颜色
    };
    divContent.onmouseout = function(){                                     //添加鼠标移出事件
        this.style.background="#FFFFFF";                                    //鼠标移开时变为原来的背景色
        this.style.color="#000000";                                         //当鼠标在关键词上时改变文字颜色
    };
    divContent.onclick=function(){                                          //添加单击事件
        document.getElementById("suggest").value=this.innerHTML;
        document.getElementById("toolBox").style.visibility="hidden";
    };
    contentObj.appendChild(divContent);                                     //将对象添加到 contentObj 对象中
}
```

（7）编写 ctrlMove()方法，用于通过↑键和↓键选择所需列表项。在该方法中，需要判断全局变量 size 是否大于 0，只有大于 0 时，才需要实现通过↑键和↓键选择所需列表项的功能。ctrlMove()方法的具体代码如下：

```
function ctrlMove(){
    if(size>0 ){
        var contentObj = document.getElementById("content");                //获取 content 对象
        size=contentObj.childNodes.length;                                  //设置条目数
        switch(event.keyCode){
        case 13:
            document.getElementById("toolBox").style.visibility="hidden";   //隐藏提示框
            size=0;
            return false;
            break;
        case 38://向上方向键
                if(curSize==-1){
                    curSize=size-1;
                }else if(curSize>0){
                    curSize--;
                }
            break;
        case 40://向下方向键
            if(curSize==-1){
                curSize=0;
            }else if(curSize<size-1){
                curSize++;
            }
            break;
        }
        if(event.keyCode==38 || event.keyCode==40){
            contentObj.childNodes[curSize].style.background="#CE83FF";      //当鼠标在关键词上时改变背景色
            contentObj.childNodes[curSize].style.color="#FFFFFF";           //当鼠标在关键词上时改变文字颜色
            document.getElementById("suggest").value=contentObj.childNodes[curSize].innerHTML;
            for(var i=0;i<size;i++){
```

```
                    if(i!=curSize){
                        contentObj.childNodes[i].style.background="#FFFFFF";      //当鼠标在关键词上时改变背景色
                        contentObj.childNodes[i].style.color="#000000";           //当鼠标在关键词上时改变文字颜色
                    }
                }
            }
        }
}
```

（8）实现当用户单击“搜索结果”按钮时，将所输入的关键字保存到指定的数据库表中。在保存时，需要判断该关键字是否已经存在，如果不存在，则保存该关键字，否则将不再保存。由于这部分内容比较简单，所以此处不再详细描述。

秘笈心法

本实例采用了第三方组件 dom4j 来实现在浏览器上输出保存符合条件关键字的 XML 文档。如果不使用 dom4j 组件，可以直接设置响应格式为 XML，然后输出每个 XML 节点以及节点文本内容即可。

实例 247 实现无刷新分页

光盘位置：光盘\MR\10\247

中级

实用指数：★★★★

实例说明

在 Web 应用开发中，经常需要对信息进行分页显示。本实例将介绍如何应用 Ajax 实现无刷新分页。运行本实例，如图 10.9 所示，单击信息列表下面的“首页”、“上一页”、“下一页”以及“尾页”超链接时，可以显示不同的信息，此时整个页面不刷新。



图 10.9　实现无刷新分页

关键技术

本实例主要应用 Ajax 异步提交来实现。当单击超链接时，需要将当前的页码参数值通过 Ajax 请求方法发送到服务器，在服务器中根据获得的参数值获取不同的分页数据，然后将这些分页数据以 XML 文档格式响应给客户端页面。客户端通过 Ajax 回调函数来读取 XML 文档内容，将这些内容替换页面信息列表中原来的显示内容，而不改变页面其他内容，从而实现页面的无刷新分页。

设计过程

（1）创建 request.js 文件，用于封装 Ajax 请求服务器的方法。

（2）创建商品信息列表页 index.jsp，在该页中通过<script>标签导入封装 Ajax 请求的 request.js 文件，然后在<script>标签中定义一个用于接收 Ajax 请求方法所返回的 XMLHttpRequest 对象的变量。创建 JavaScript 方法 getData()，在该方法中调用 request.js 文件中的 Ajax 请求方法请求目标服务器，关键代码如下：

```
<script language="javascript" src="js/request.js"></script>
<script language="javascript">
var request = false;                                                    //保存 XMLHttpRequest 对象的全局变量
```

```
//Ajax 请求方法
function getData(page){
	var url="GoodsServlet";						//服务器地址
	var param ="page="+page+"&nocache="+new Date().getTime();	//请求参数
	request=httpRequest("post",url,true,callbackFunc,param);	//调用 request.js 请求方法
}
```

（3）创建 Servlet 的实现类 GoodsServlet，用于获取商品信息。在该类的 doPost()方法中接收 Ajax 请求中的 page 参数值，然后设置响应正文的 MIME 类型为 XML 格式，从数据库中读取出商品信息以及分页信息添加到 XML 的节点中，最后将 XML 格式的信息输出到客户端。doPost()方法的具体代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
		throws ServletException, IOException {
	request.setCharacterEncoding("UTF-8");
	response.setContentType( "text/xml;charset=UTF-8" );//设置响应格式为 test/xml
	//设置 response.addHeader()，禁止页面缓存
	response.addHeader( "Cache-Control", "no-store,no-cache,must-revalidate" );
	response.addHeader( "Cache-Control", "post-check=0,pre-check=0" );
	response.addHeader( "Expires", "0" );
	response.addHeader( "Pragma", "no-cache" );
	PrintWriter out = response.getWriter();
	PageElement pageArgs = getPageArgs();			//获取数据分页参数
	List goodses = null;					//创建保存商品信息的数据集合
	//获取请求对象的分页参数
	int page=1;
	if(null!=request.getParameter("page")){
		page = Integer.parseInt(request.getParameter("page"));
	}
	pageArgs.setPageCount(page);
	out.println( "<?xml version=\"1.0\" encoding=\"utf-8\"?>" );
	out.println( "<goodses>" );
	out.println( "<pageElement>" );
	out.println( "<pageNum>" +pageArgs.getPageNum()+"</pageNum>");
	out.println( "<maxPage>" +pageArgs.getMaxPage()+"</maxPage>");
	out.println( "<nextPage>" +pageArgs.getNextPage()+"</nextPage>");
	out.println( "<prePage>" +pageArgs.getPrePage()+"</prePage>");
	out.println( "</pageElement>" );
	try{
		goodses = getGoods(page);
	}catch(Exception e){
		e.printStackTrace();
	}
	for(int i=0;i<goodses.size();i++){
		 CommodityForm commodity = (CommodityForm) goodses.get(i);
		 out.println( "<goods>" );
		 out.println( "<id>"+commodity.getId()+"</id>" );
		 out.println( "<goodsName>"+commodity.getGoodsName()+"</goodsName>" );
		 out.println( "<introduce>"+commodity.getIntroduce()+"</introduce>" );
		 out.println( "<Price>"+commodity.getPrice()+"</Price>" );
		 out.println( "</goods>" );
	}
	out.println( "</goodses>" );
	out.close();
}
```

（4）创建 PageElement 类，用于封装计算分页的参数，关键代码如下：

```
public class PageElement {
	private int pageNum;				//当前页
	private int pageSize;				//页面显示数据的条数
	private long maxPage;				//总页数
	private int prePage;				//上一页
	private int nextPage;				//下一页
	public void setPageCount(int pageNum) {
		this.pageNum = pageNum;
		//设置上一页的页码
		prePage=pageNum-1<=1?1:pageNum-1;
		//设置下一页的页码
		nextPage=(int)(pageNum + 1 >= maxPage ? maxPage : pageNum + 1);
	}
```

```
…        //此处省略了其他属性的 getXXX()和 setXXX()方法
}
```

（5）在 GoodsServlet 类中编写获取分页参数信息的 getPageArgs()方法，具体代码如下：

```
public PageElement getPageArgs() {
        PageElement pag = null;                    //声明分页参数对象
        Statement stmt = null;
        ResultSet rs = null;
        ConnDB conn = new ConnDB();                //连接数据库
        try {
            //声明查询总数据量的 SQL 语句
            String sql = "SELECT count(*) FROM tb_goods";
            rs = conn.executeQuery(sql);           //执行 SQL 查询
            if (rs.next()) {
                int count = rs.getInt(1);          //获取查询结果
                pag = new PageElement();           //初始化分页参数对象
                pag.setPageSize(pagesize);         //设置分页大小
                pag.setMaxPage((count + pagesize - 1) / pagesize);//设置最大页码
                pag.setPageCount(count);           //设置当前页码
            }
        } catch (SQLException ex) {
            ex.getMessage();
        } finally {
            conn.close();
        }
        return pag;
}
```

（6）在 GoodsServlet 类中编写从数据库读取指定商品信息的 getGoods()方法，具体代码如下：

```
public List getGoods(final int page) throws Exception {
        List list = new ArrayList();               //创建保存分页数据的集合对象
        ConnDB conn = new ConnDB();                //连接数据库
        CommodityForm f =null;
        int firstResult = (page-1) * pagesize;
        try {
            //定义分页查询的 SQL 语句
            String sql= "select * from tb_goods order by id limit "+firstResult+","+pagesize;
            ResultSet rs = conn.executeQuery(sql); //获取查询结果
            while (rs.next()) {                    //遍历查询结果集
                f = new CommodityForm();           //创建分页对象
                  f.setId(rs.getInt("id"));
                f.setGoodsName(rs.getString("name"));
                f.setIntroduce(rs.getString("introduce"));
                f.setPrice(rs.getFloat("price"));
                list.add(f);                       //添加分页数据对象到 List 集合
            }
        } catch (SQLException ex) {
           ex.printStackTrace();
        } finally {
            conn.close();
        }
        return list;
}
```

（7）编写完以上 GoodsServlet 类中的几个方法后，在 index.jsp 中需要编写 Ajax 回调函数 callbackFunc()，在回调函数中读取 Ajax 请求 GoodsServlet 所响应的 XML 数据，并将其添加到指定<span>标签中。callbackFunc()的具体代码如下：

```
function callbackFunc(){
 if(request.readyState == 4){
      if(request.status == 200){
         var doc = request.responseXML;
         var pageNum = doc.getElementsByTagName("pageNum")[0].firstChild.data;
         var maxPage= doc.getElementsByTagName("maxPage")[0].firstChild.data;
         var prePage = doc.getElementsByTagName("prePage")[0].firstChild.data;
         var nextPage = doc.getElementsByTagName("nextPage")[0].firstChild.data;
         var goodses = doc.getElementsByTagName("goods");
         var innerHTML = "";
         if((goodses!=null)&&(goodses.length!=0)) {
             innerHTML+="<table  width='96%'  border='1'  cellpadding='0'  cellspacing='0'  bordercolor='#FFFFFF'  bordercolordark='#FFFFFF'
```

```
bordercolorlight='CFCFCF'>";
                    innerHTML+="<tr align='center' height='30'><td>商品名称</td>";
                    innerHTML+="<td>简介</td>";
                    innerHTML+="<td>单价</td></tr>";
                    for(var i=0;i<goodses.length;i++) {
                        var goods = goodses[i];
                        var goodsName = goods.childNodes[1].firstChild.data;
                        var introduce = goods.childNodes[2].firstChild.data;
                        var price = goods.childNodes[3].firstChild.data;
                        innerHTML += "<tr height='30'>";
                        innerHTML += "<td align='left'> "+goodsName+"</td>";
                        innerHTML += "<td align='left'> "+introduce+"</td>";
                        innerHTML += "<td align='left'> "+price+"</td>";
                        innerHTML += "</tr>";
                    }
                    innerHTML+="<tr height='30'><td align='center' colspan='6'>";
                    innerHTML +="["+ pageNum+"/"+maxPage+"]  ";
                    innerHTML+="<a href=\"javascript:void(0)\" onClick=\"getData(1)\">[首  页]</a> ";
                    innerHTML+= "<a href=\"javascript:void(0)\" onClick=\"getData("+prePage+")\">[上一页]</a> ";
                    innerHTML+= "<a href=\"javascript:void(0)\" onClick=\"getData("+nextPage+")\">[下一页]</a> ";
                    innerHTML +="<a href=\"javascript:void(0)\" onClick=\"getData("+maxPage+")\">[尾页]</a>";
                    innerHTML+="</td></tr>\n";
                    innerHTML+="</table>\n";
                }else {
                    innerHTML += "暂时没有任何数据";
                }
                document.getElementById("goodsList").innerHTML = innerHTML;
                request = false;//此处必须设置 XMLHttpRequest 对象的变量为初始状态
            }
        }
}
```

（8）在 index.jsp 页的指定位置添加显示商品列表信息的<span>标签，并设置该标签的 id 属性为 goodsList，关键代码如下：

```
<span id="goodsList"></span>
```

（9）在页面加载后就应该请求服务器，读取出初始商品信息，关键代码如下：

```
window.onload=function(){
        getData(1);
}
```

秘笈心法

本实例在获得服务器回传的 XML 数据时，应用了 JavaScript 中的 DOM 对象。在 DOM 对象中，HTML 文档各个节点被视为各种类型的 Node 对象，并且将 HTML 文档表示为 Node 对象的树，对于任何一个树形结构来说，最常做的就是遍历树。DOM 中可以通过 Node 对象的 parentNode 属性、firstChild 属性、lastChild 属性、nextChild 属性、previousSibling 属性和 nextSibling 属性来遍历文档树。Node 对象的常用属性的具体作用如表 10.2 所示。

表 10.2　Node 对象的属性

属　　性	类　　型	描　　述
parentNode	Node	节点的父节点，没有父节点时为 null
childNodes	NodeList	节点的所有子节点的 NodeList
firstChild	Node	节点的第一个子节点，没有则为 null
lastChild	Node	节点的最后一个子节点，没有则为 null
previousSibling	Node	节点的上一个节点，没有则为 null
nextChild	Node	节点的下一个节点，没有则为 null
nodeName	String	节点名
nodeValue	String	节点值
nodeType	short	表示节点类型的整型常量

实例 248 实时弹出气泡提示窗口

光盘位置：光盘\MR\10\248

高级

实用指数：★★★★☆

实例说明

在浏览网站时，有些网站会在窗口的右下角弹出气泡提示窗口，显示网站的最新公告或广告。本实例将介绍如何应用 Ajax 实现实时弹出气泡提示窗口。运行本实例，系统会自动判断当天是否有最新的新闻。如果有，则弹出一个气泡提示窗口显示该新闻，如图 10.10 所示。在该窗口中单击新闻标题超链接，将弹出新窗口显示新闻的详细内容。当用户查看该新闻的详细内容或是单击“×”按钮后，将不再弹出气泡提示窗口，除非又有新的新闻发布。



图 10.10　实时弹出气泡提示窗口

关键技术

本实例的关键之处是如何弹出一个气泡提示窗口。在 Web 应用中，可以通过 HTML DOM 中 window 对象的 createPopup()方法实现创建一个气泡提示窗口，下面将对该方法进行详细介绍。

createPopup()方法用于创建一个弹出窗口（pop-up），该窗口和普通窗口一样具有标准窗口所拥有的属性和事件。但是通过该方法创建的窗口对象不能使用父窗口的 CSS 风格，需要手工重写。createPopup()方法的基本语法如下：

```
oPopup = window.createPopup()
```

在该方法中，返回值为 oPopup Object 对象，也就是返回弹出窗口（pop-up）对象。

设计过程

（1）创建 request.js 文件，用于封装 Ajax 请求服务器的方法。

（2）创建 index.jsp 页，在该页的<script>标签中导入 request.js 文件。编写实例化 Ajax 对象的方法 loadBubbleTip()，用于向服务器发送请求，获取当天最新的一条新闻，关键代码如下：

```
var bbsid=0;                                                    //当前公告 ID
var request = false;                                            //Ajax 请求函数返回的 XMLHttpRequest 对象
function loadBubbleTip(){
        var url="BbsServlet";                                   //服务器地址
        var param ="action=getBbs&nocache=" + new Date().getTime();    //请求参数
```

```
        request=httpRequest("post",url,true,callback,param);                    //调用请求方法
}
```

（3）创建用于处理请求的 Servlet 实现类 BbsServlet，在该类的 doPost()方法中获取 action 参数的值，根据 action 参数的值来调用 getBbs()方法，从数据库中获取当天的最新的一条新闻。doPost()方法的具体代码如下：

```
protected void doPost(HttpServletRequest request,HttpServletResponse response)
        throws ServletException, IOException {
        String action = request.getParameter("action");                         //获取 action 参数的值
        if ("getBbs".equals(action)) {
                getBbs(request, response);                                       //查询最新的新闻的方法
        }
}
```

（4）在 BbsServlet 中编写用于查询最新新闻信息的 getBbs()方法。在该方法中首先获取当天发表的最后一条新闻，也就是最新的一条新闻，然后将获取的新闻信息以 XML 格式连接成一个字符串，再判断当前 cookie 中是否存在包括查询到的新闻的 ID 号的 cookie，也就是判断该新闻是否已读。如果已读，则设置输出字符串为根节点的起始标记，最后将组合后的字符串输出到浏览器。getBbs()方法的具体代码如下：

```
private void getBbs(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        //查询最新的新闻信息
        ConnDB conn = new ConnDB();
        Date date=new Date();
        String now=new java.sql.Date(date.getTime()).toString();                //获取 yyyy-MM-dd 格式的日期
        String sql = "SELECT * FROM tb_bbs WHERE sendTime between '"+now+" 00:00:00' AND '"+now+"
23:59:59' ORDER BY sendTime DESC LIMIT 1";
        ResultSet rs = conn.executeQuery(sql);                                   //执行查询语句
        int id=0;
        //设置响应的类型和编码方式
        response.setContentType("text/xml;charset=UTF-8");
        response.setHeader("Cache-Control", "no-cache");
        PrintWriter out = response.getWriter();
        String output = "<bbs>";
        //将信息输出在 XML 文件内
        SimpleDateFormat m = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");       //设置日期时间的格式
        try {
                while (rs.next()) {
                        id=rs.getInt(1);                                         //获取新闻 ID
                        output += "<id>" + id + "</id>";
                        output += "<title>" + rs.getString(2) + "</title>";
                        output += "<time>"+ m.format(java.text.DateFormat.getDateTimeInstance()
                                        .parse(rs.getString(4))) + "</time>";
                }
        } catch (Exception e) {
                e.printStackTrace();
        }
        //判断当前 cookie 中是否存在该 ID 号的 cookie，也就是判断该新闻是否已读
        Cookie cookies[]=request.getCookies();                                   //读取全部 cookie
        for(int i=0;i<cookies.length;i++){
                if("mrBbsRemind".equals(cookies[i].getName()) && ("bbsid"+id).equals(cookies[i]
                                .getValue())){
                        output = "<bbs>";
                        break;                                                   //跳出循环
                }
        }
        output += "</bbs>";
        out.println(output);                                                     //输出组合后的字符串
        out.close();                                                             //关闭输出流
}
```

（5）在 index.jsp 文件中，编写获取最新新闻信息的 Ajax 回调函数。在回调函数中，首先获取 XML 格式的新闻信息，然后从该新闻信息中获取新闻 ID、新闻标题、新闻发布时间并保存到相应的变量中，再通过实例化气泡提示类创建一个气泡提示窗口，最后设置该气泡提示窗口的相应属性，并且添加查看新闻详细信息命令，关键代码如下：

```
function callback() {
if(request.readyState == 4){
```

```
        if(request.status == 200){
            var doc = request.responseXML; //获取 XML 格式的新闻信息
                if(doc.getElementsByTagName("title").length>0){
                    //获取 XML 文档中的 id 节点的第 1 个子节点的值
                    var id = doc.getElementsByTagName("id")[0].firstChild.nodeValue;
                    bbsid=id;
                    //获取 XML 文档中的 title 节点的第 1 个子节点的值
                    var content = doc.getElementsByTagName("title")[0].firstChild.nodeValue;
                    //获取 XML 文件中的 time 节点的第 1 个子节点的值
                    var foot1= doc.getElementsByTagName("time")[0].firstChild.nodeValue;
                    var remindMessage = new PopBubble(307,170," ",content,foot1);
                    //设置弹出窗口的左边、右边、顶边和底边框的位置
                    remindMessage.box(null,null,null,screen.height-30);
                    remindMessage.speed = 10;                //设置窗口的弹出速度
                    remindMessage.step = 2;                  //设置窗口的弹出步幅
                    remindMessage.show();                    //弹出窗口
                    PopBubble.prototype.oncommand = function(){
                    window.open("BbsServlet?action=getDetail&id="+id,"","width=513,height=567,scrollbars=1");
                    this.close = true;
                    this.hide();                             //收缩窗口
                }
            }
            request = false;
        }
}
}
```

（6）在页面载入后，调用 loadBubbleTip()方法弹出气泡提示窗口，并且设置每隔 1 分钟调用一次 loadBubbleTip()方法，关键代码如下：

```
window.onload=function(){
        loadBubbleTip();
        window.setInterval(loadBubbleTip,1000*60);
}
```

（7）编写气泡提示类，并将全部代码保存在一个名为 remind.js 的文件中，然后在弹出的气泡提示窗口的 index.jsp 页面中引用 remind.js 文件。

创建一个用于弹出气泡提示的类，名称为 PopBubble，并且在该类中定义相应的属性，具体代码如下：

```
function PopBubble(width,height,title,content,foot){
      this.content    = content;                      //提示内容
      this.title= title;                              //提示标题
      this.foot= foot;                                //页脚内容
      this.width = width?width:307;                   //设置弹出窗口的宽度
      this.height = height?height:170;                //设置弹出窗口的高度
      this.timeout= 300;                              //设置窗口的停留时间
      this.speed = 10;                                //设置窗口的弹出速度
      this.step = 1;                                  //设置窗口的弹出步幅
      this.right = screen.width -1;                   //设置窗口右边的位置
      this.bottom = screen.height;                    //设置窗口底边的位置
      this.left = this.right - this.width;            //设置窗口左边的位置
      this.top = this.bottom - this.height;           //设置窗口顶边的位置
      this.timer = 0;                                 //计时器
      this.pause = false;                             //标记是否停止收缩
      this.close = false;                             //标记是否关闭
      this.autoHide = true;                           //标记是否自动收缩
}
```

为弹出气泡提示类编写 hide()方法，该方法用于收缩弹出窗口，具体代码如下：

```
PopBubble.prototype.hide = function(){
            //设置弹出窗口的高度
        var offset    = this.height>this.bottom-this.top?this.height:this.bottom-this.top;
        var obj    = this;
        if(this.timer>0){
            window.clearInterval(obj.timer);
        }
        var fun = function(){
            if(obj.pause==false||obj.close){                //判断是否弹出窗口
                var x = obj.left;                           //获取窗口左边框的位置
```

```
            var y = 0;
            var width = obj.width;                    //获取窗口的宽度
              //获取窗口的高度
            var height = 0;
            if(obj.offset>0){
                height = obj.offset;
            }
            y   = obj.bottom - height;                //获取窗口顶边框的位置
            if(y>=obj.bottom){
                window.clearInterval(obj.timer);
                obj.Pop.hide();                       //收缩窗口
            } else {
                obj.offset = obj.offset - obj.step;
            }
            obj.Pop.show(x,y,width,height);           //弹出窗口
        }
    }
    this.timer = window.setInterval(fun,this.speed);
}
```

为弹出气泡提示类编写 show()方法，该方法用于显示带声音提示的弹出窗口，具体代码如下：

```
PopBubble.prototype.show = function(){
    var oPopup = window.createPopup();                //创建一个顶层窗口对象
    this.Pop = oPopup;
    var w = this.width;
    var h = this.height;
    this.offset   = 0;
    var str = "<div style='border:#1D84D3 2px solid;z-index: 99999; left: 0px; width: " + w + "px; position: absolute; top: 0px; height: " + h + "px;'>"
        str += "<table style='background-image:url(images/pop_title.jpg);background-repeat:no-repeat;   cellSpacing=0 cellPadding=0 width='100%' border=0>"
        str += "<tr>"
        str += "<td width=30 height=33></td>"
        str += "<td style='padding-left: 4px; font-weight: normal; font-size: 12px; color: #331C09; padding-top: 4px' vAlign=center width='100%'>" + this.title + "</td>"
        str += "<td style='padding-right: 2px; ' valign=center align=right width=19>"
        str += "<span title=关闭  style='font-weight: bold; font-size: 12px; cursor: hand; color: #FF4200; margin-right: 4px' id='btn_Close' >×</span></td>"
        str += "</tr>"
        str += "<tr>"
        str += "<td style='padding: 1px' colSpan=3 height=" + (h-28) + ">"
        str += "<div style='background-color: #feffff1;padding: 8px;  font-size: 12px;  width: 100%; height: 100%'><a href='javascript:void(0)' hidefocus=true id='aLink' style='color:#990000'>" + this.content + "</a><br><br>"
 str += "<div style='word-break: break-all' align='right' style='color:#6C6B68'>" + this.foot +
     "<embed id='soundControl' src='images/Windows.wav' mastersound hidden='true' loop='0' autostart='true'></embed>"+ "</div>"
        str += "</div>"
        str += "</td>"
        str += "</tr>"
        str += "</table>"
        str += "</div>"
      //将设置好的 div  添加到顶层窗口的 body 中
    oPopup.document.body.innerHTML = str;
    var obj   = this;
      //添加顶层窗口的鼠标悬停事件，在事件中设置窗口不收缩
    oPopup.document.body.onmouseover = function(){obj.pause=true;}
      //添加顶层窗口的鼠标移除事件，在事件中设置窗口收缩
    oPopup.document.body.onmouseout = function(){obj.pause=false;}
    var fun = function(){
    var x = obj.left;                                 //获取窗口左边框的位置
    var y = 0;
    var width = obj.width;                            //获取窗口的宽度
      //获取窗口的高度
    var height      = obj.height;
    if(obj.offset>obj.height){
          height = obj.height;
     } else {
           height = obj.offset;
     }
      //获取窗口顶边框的位置
    y   = obj.bottom - obj.offset;
```

```
    if(y<=obj.top){
        obj.timeout--;
        if(obj.timeout==0){
        window.clearInterval(obj.timer);
            if(obj.autoHide){
                obj.hide();                              //收缩窗口
            }
        }
    } else {
        obj.offset = obj.offset + obj.step;
    }
    obj.Pop.show(x,y,width,height);                     //弹出窗口
    }
    this.timer = window.setInterval(fun,this.speed)      //间隔 speed 时间调用 fun 函数弹出窗口
    //单击“关闭”按钮时需要请求服务器执行操作
    oPopup.document.getElementById("btn_Close").onclick = function(){
        var url="BbsServlet";
        var param ="action=changeFlag&id="+bbsid+"&nocache=" + new Date().getTime();
        request=httpRequest("post",url,true,callback1,param);   //调用请求方法
        obj.close = true;
        obj.hide();                                      //收缩窗口
    }
    //单击超链接时执行的操作
   oPopup.document.getElementById("aLink").onclick = function(){
        obj.oncommand();                                 //调用 oncommand()函数打开查看超链接详细信息的窗口
    }
}
```

注意：在上面的代码中，加粗的代码用于通过 Ajax 实现单击“关闭”按钮时写入 cookie，保证该条新闻不再对当前用户显示。

编写设置弹出窗口的左边、右边、顶边和底边框的位置的方法，具体代码如下：

```
PopBubble.prototype.box = function(left,right,top,bottom){
    try {
        this.left = left !=null?left:this.right-this.width;
        this.right = right!=null?right:this.left +this.width;
        this.bottom    = bottom!=null?(bottom>screen.height?screen.height:bottom):
                                    screen.height;
        this.top = top !=null?top:this.bottom - this.height;
    } catch(e){}
}
```

秘笈心法

在实现本实例时，采用了 Cookie 保存已经读取或关闭的新闻 ID。这样，在获取新闻信息时，先判断新闻 ID 是否存在于该用户的 cookie 中，如果存在，就不对该用户显示这条新闻，否则才弹出气泡提示显示该新闻。这就能保证在用户查看或关闭新闻后，这条新闻信息将不再对该用户所显示。

10.3　动态加载数据

为了提高页面的加载速度，可以对一些数据应用动态加载。下面通过几个具体的例子介绍如何应用 Ajax 实现动态加载数据。

实例 249	实时显示最新商品及报价 光盘位置：光盘\MR\10\249	初级 实用指数：★★★☆

实例说明

本实例将介绍如何应用 Ajax 实现实时显示最新商品及报价。运行本实例，在页面的中间位置将显示“最新

商品及报价”栏目，如图 10.11 所示，每隔 10 分钟从数据库中获取一次最新商品信息及报价，保证所看到的数据为最新数据。



图 10.11　实时显示最新商品及报价

关键技术

本实例主要是利用 Ajax 异步提交请求的方式来定时读取数据库中最新的商品信息。实现定时读取的关键是通过 JavaScript 中 window 对象的 setInterval()方法，通过该方法来调用 Ajax 请求服务器的方法，并且设置每隔 10 分钟调用一次，即可实现实时显示最新商品及报价。

设计过程

（1）创建 request.js 文件，用于封装 Ajax 请求服务器的方法。

（2）创建 index.jsp 页，在该页面中导入 request.js 文件，应用 DIV+CSS 进行布局，然后在<script>标签中编写实例化 Ajax 对象的方法 getGoods()，用于向服务器发送请求，获取最新的商品信息及报价，关键代码如下：

```
<script src="js/request.js"></script>
<script language="javascript">
var request =false; //创建用于保存 XMLHttpRequest 对象的变量
function getGoods() {
        var url="GoodsServlet";                                          //服务器地址
        var param ="action=getGoods&nocache=" + new Date().getTime();   //请求参数
        request=httpRequest("post",url,true,callbackFunc,param);         //调用请求方法
}
```

（3）创建 Servlet 的实现类 GoodsServlet，在该类的 doPost()方法中获取 action 参数的值，并且判断 action 参数的值是否等于 getGoods，如果等于，则调用 getGoods()方法从数据库中获取最新的商品信息及报价。doPost()方法的具体代码如下：

```
protected void doPost(HttpServletRequest request,HttpServletResponse response)
        throws ServletException, IOException {
        String action = request.getParameter("action");
        if ("getGoods".equals(action)) {
                getGoods(request, response);
        }
}
```

（4）在 GoodsServlet 中编写 getGoods()方法，在该方法中首先设置响应类型为 XML，并且创建 XML 文档对象及根节点，然后从数据库中获取最新的商品信息及报价，再将获取的商品信息按照指定的格式写入到 XML 文档对象中，最后将 XML 文档输出到浏览器。getGoods()方法的具体代码如下：

```
private void getGoods(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException{
        request.setCharacterEncoding("UTF-8");
        response.setContentType( "text/xml;charset=UTF-8" );              //设置响应格式为 XML
        //禁止页面缓存
        response.addHeader( "Cache-Control", "no-store,no-cache,must-revalidate" );
```

```
        response.addHeader( "Cache-Control", "post-check=0,pre-check=0" );
        response.addHeader( "Expires", "0" );
        response.addHeader( "Pragma", "no-cache" );
        PrintWriter out = response.getWriter();
        ConnDB conn = new ConnDB();
        String sql = "SELECT * FROM tb_goods ORDER BY refreshTime DESC";
        ResultSet rs = conn.executeQuery(sql);                        //查询数据库商品信息
        out.println("<goodses>");                                      //XML 根节点
        try{
                while(rs.next()){                                      //循环结果集，将读取出的数据添加到 XML 子节点中
                        out.println("<goods>");
                        out.println("<name>"+rs.getString(2)+"</name>");
                        out.println("<price>"+rs.getFloat(3)+"</price>");
                        out.println("<picture>images/goods/"+rs.getString(4)+"</picture>");
                        out.println("</goods>");
                }
        }catch(Exception ex){
                ex.printStackTrace();
        }
        out.println("</goodses>");
        out.close();
}
```

（5）在 index.jsp 页面的合适位置添加一个显示最新商品及报价的<div>标记，id 为 goodsList，关键代码如下：

```
<div id="goodsList">正在获取商品信息...</div>
```

（6）在 index.jsp 文件中编写 Ajax 回调函数。首先需要获取 XML 格式的商品信息，然后通过循环将获取的商品信息连接成指定格式的字符，最后将其显示到<div>标记中，关键代码如下：

```
function callbackFunc() {
        if(request.readyState == 4){
                if(request.status == 200){
                        var doc = request.responseXML;                 //获取 XML 格式的商品信息
                        var goodses = doc.getElementsByTagName("goods");  //获取 goods 节点对应的对象
                        var innerHTML = "";
                        if ((goodses != null) && (goodses.length != 0)) {
                                innerHTML += "<ul>";
                                for (i = 0; i < goodses.length; i++) {   //通过循环将获取每个 goods 节点的值
                                        var goods = goodses[i];
                                        var goodsName = goods.childNodes[0].firstChild.data; //获取商品名称
                                        var price = goods.childNodes[1].firstChild.data;     //获取当前价格
                                        var picture = goods.childNodes[2].firstChild.data;   //获取商品图片
                                        innerHTML += "<li><dl>";
                                        innerHTML += "<dt><img src='" + picture + "'></dt>";
                                        innerHTML += "<dd>" + goodsName + "</dd>";
                                        innerHTML += "<dd>当前价格：<font color='red'>" + price+ "</font>（元）</dd>";
                                        innerHTML += "</dl></li>";
                                }
                                innerHTML += "</ul>";
                                innerHTML += innerHTML; //为了实现走马灯效果，所以添加这句代码
                        } else {
                                innerHTML = "暂无最新商品及报价！ ";
                        }
                        document.getElementById("goodsList").innerHTML = innerHTML;
                        request = false;
                }
        }
}
```

（7）为了实现实时获取最新的商品信息及报价，还需要在 index.jsp 文件中添加以下的 JavaScript 代码，从而实现当页面载入完毕后，先调用 getGoods()方法获取商品信息，然后再设置每隔 10 分钟获取一次商品信息，关键代码如下：

```
window.onload = function() {
        getGoods();
        window.setInterval("getGoods()", 600000);                     //每隔 10 分钟调用一次 getGoods()方法
}
```

秘笈心法

由于在 Servlet 中回传给客户端的 XML 文档中的商品信息是由多个<goods>组成的节点，所以在 Ajax 回调函数中获取 XML 节点时，应该通过循环来获取每个<goods>节点中的数据。

实例 250　实时显示聊天内容

光盘位置：光盘\MR\10\250

初级

实用指数：★★★★

实例说明

应用 Ajax 可以实现在不刷新页面的情况下实时显示聊天内容。运行本实例，在页面中将显示最新的聊天内容，如图 10.12 所示，当用户输入昵称及说话内容，单击“发送”按钮后，将发送聊天内容，并显示到上方的聊天内容列表中。

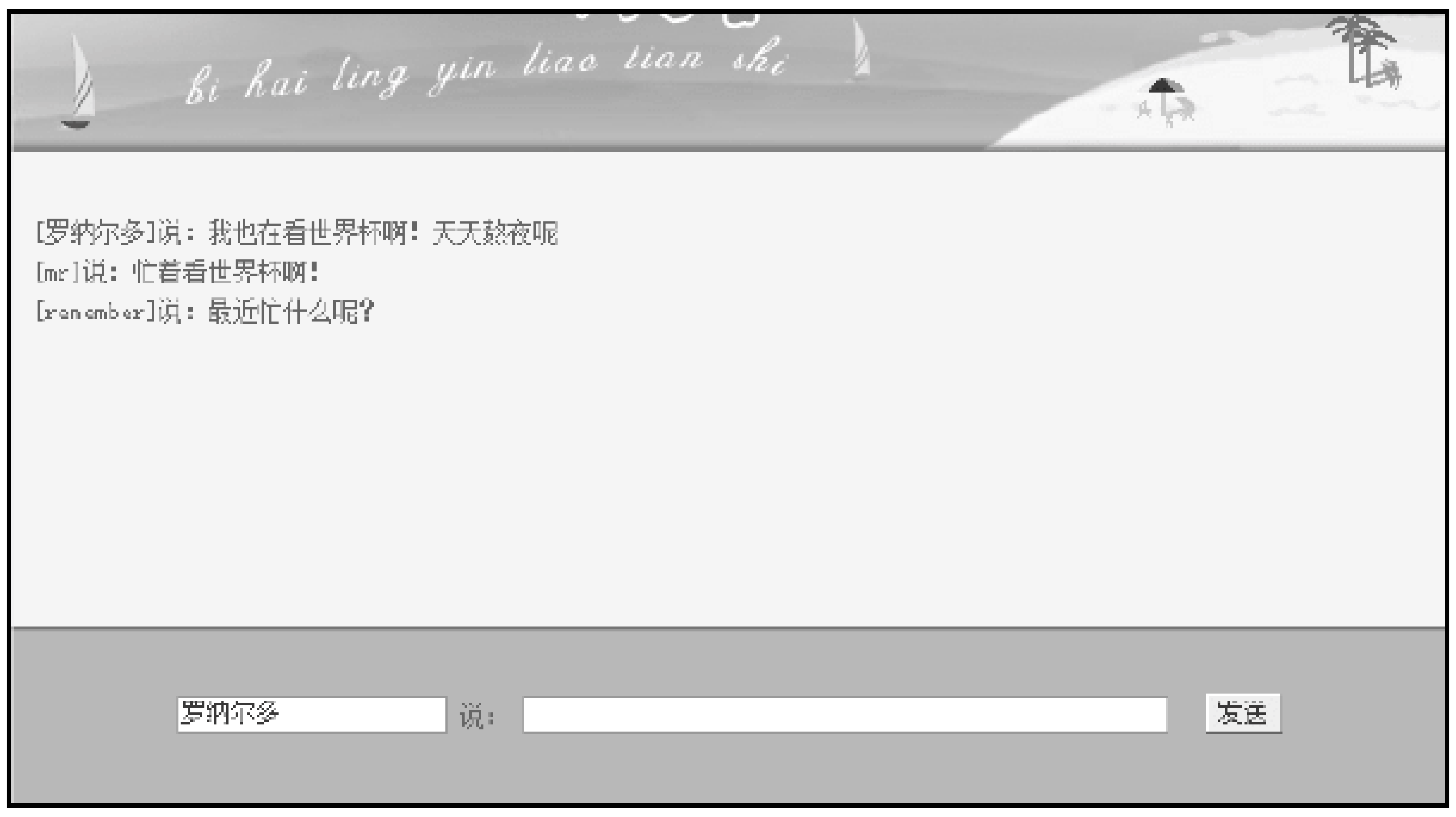


图 10.12　实时显示聊天内容

关键技术

本实例主要应用 Ajax 异步提交请求技术实现不刷新页面的同时发送聊天信息并实时显示最新的聊天信息。

设计过程

（1）创建 request.js 文件，用于封装 Ajax 请求服务器的方法。

（2）创建聊天页面 index.jsp，在该页面中首先导入 request.js 文件，然后在<script>标签中编写实例化 Ajax 对象的方法 send()，用于向服务器发送请求，关键代码如下：

```
<script language="javascript" src="js/request.js"></script>
<script language="javascript">
var request =false;
function send(){                                              //验证聊天信息并发送
    var user = document.getElementById("user").value;
    var speak = document.getElementById("speak").value;
    if(user==""){
        alert("请输入您的昵称！");
        return false;
    }
    if(speak==""){
        alert("发送信息不可以为空！");
        document.getElementById("speak").focus();
        return false;
    }
    document.getElementById("speak").value="";                //清空文本框的值
    document.getElementById("speak").focus();                 //让文本框获得焦点
   var url="Chat";                                            //服务器地址
    var param ="action=send&user="+user+"&speak="+speak;      //请求参数
```

```
        request=httpRequest("post",url,true,callback,param);                    //调用请求方法
}
```

（3）创建 Servlet 实现类 Chat，在该类中编写 send()方法，将聊天信息保存到 application 对象中。send()方法的具体代码如下：

```
public void send(HttpServletRequest request,HttpServletResponse response)
        throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        response.setCharacterEncoding("UTF-8");
        ServletContext application=getServletContext();                        //获取 application 对象
        String user=request.getParameter("user");                              //获取用户昵称
        String speak=request.getParameter("speak");                            //获取说话内容
        user = java.net.URLDecoder.decode(user, "UTF-8");
        speak = java.net.URLDecoder.decode(speak, "UTF-8");
        Vector<String> v=null;
        String message="["+user+"]说："+speak;                                 //组合说话内容
        if(null==application.getAttribute("message")){
                v=new Vector<String>();
        }else{
                v=(Vector<String>)application.getAttribute("message");
        }
        v.add(message);
        application.setAttribute("message", v);                                //将聊天内容保存到 application 中
       Random random = new Random();
        request.getRequestDispatcher("Chat?action=get&nocache=" + random.nextInt(10000))
.forward(request, response);
}
```

（4）在 index.jsp 页中，编写实例化 Ajax 对象的方法 getContent()，向服务器发送请求获取聊天内容。并且编写 Ajax 回调函数读取服务器返回的聊天内容，关键代码如下：

```
function getContent(){
        var url="Chat";                                                        //服务器地址
        var param ="action=get&nocache="+new Date().getTime();                 //请求参数
        request=httpRequest("post",url,true,callback_content,param);           //调用请求方法
}
//Ajax 回调函数
function callback_content(){
 if(request.readyState == 4){
        if(request.status == 200){
                document.getElementById("content").innerHTML=request.responseText;  //显示聊天内容
                request =false;
        }
 }
}
```

（5）在 Chat 类中编写 get()方法获取全部聊天信息。get()方法的具体代码如下：

```
public void get(HttpServletRequest request,HttpServletResponse response)
        throws ServletException,IOException{
        ServletContext application=getServletContext();                        //获取 application 对象
        String msg="";
        if(null!=application.getAttribute("message")){
                Vector<String> v_temp=(Vector<String>)application.getAttribute("message");
                for(int i=v_temp.size()-1;i>=0;i--){
                        msg=msg+"<br>"+v_temp.get(i);
                }
        }else{
                msg="欢迎光临本聊天室！";
        }
       request.setAttribute("msg", msg);
       request.getRequestDispatcher("content.jsp").forward(request, response);
}
```

（6）编写 content.jsp 文件，在该文件中应用 EL 表达式输出聊天内容，具体代码如下：

```
<%@page contentType="text/html" pageEncoding="GBK" %>
${msg}
```

（7）为了实现实时显示最新的聊天内容，还需要在 index.jsp 页中添加以下代码：

```
window.setInterval("getContent();",1000);
window.onload=function(){
```

```
    getContent();                //当页面载入后调用 Ajax 获取聊天内容
}
```

（8）在“发送”按钮的 onClick 事件中调用 send()方法，发送聊天信息，关键代码如下：

```
<input type="button" value="发送" onClick="send()">
```

秘笈心法

本实例在实现时，在 Servlet 中应用到了 ServletContext 对象保存用户发送的聊天内容，在 ServletContext 对象中保存的数据是共享数据，整个 Web 应用程序都可以访问到该对象中保存的数据，所以用 ServletContext 对象保存聊天内容，每个聊天用户都能够访问。

实例 251　实现快速浏览

光盘位置：光盘\MR\10\251

高级　实用指数：★★★★

实例说明

所谓快速浏览，就是当鼠标指针移动到指定区域时，会动态加载显示相关数据，这些动态加载的数据是通过 Ajax 异步请求读取出来的，而不是网页中的静态信息。本实例将介绍如何应用 Ajax 来实现这样的效果。运行本实例，如图 10.13 所示，当鼠标指针移动到某件商品所在区域时，将弹出窗口显示该商品的详细信息。



图 10.13　实现快速浏览

关键技术

在实现本实例时，除了应用 Ajax 技术外，还需要应用 DOM 来生成一个准确的偏移量，并根据该偏移量放置工具提示。主要应用 HTML 元素的 offsetLeft 属性和 offsetTop 属性，offsetLeft 属性用于返回当前元素相对于版面或其 offsetparent 属性返回对象的左边缘的偏移量，offsetTop 属性用于返回当前元素相对于版面或其 offsetparent 属性返回对象的顶端的偏移量。

设计过程

（1）创建 request.js 文件，用于封装 Ajax 请求服务器的方法。

（2）创建显示商品信息的页面 index.jsp，在该页的<script>标签中导入 request.js 文件，然后在页面中添加显示商品信息的<div>标签，并在<div>中添加分栏显示商品概要信息的表格，然后在表格的下方添加一个用于显示工具提示的<div>标签，关键代码如下：

```
<div id="toolTip" style=" background-image:url(images/tip.gif); width:206px; height:155px; display:none; position:absolute; z-index:100;">
        <table width="100%" height="138" border="0" cellpadding="0" cellspacing="0">
            <tr>
              <td width="20%" height="28"> </td>
              <td width="71%" valign="middle"> </td>
              <td width="9%" valign="middle"> </td>
            </tr>
            <tr>
              <td> </td>
              <td valign="top" id="goodsIntroduce">介绍</td>
              <td valign="top"> </td>
            </tr>
        </table>
</div>
```

（3）在 index.jsp 页的<script>标签中，编写实例化 Ajax 对象的方法 getData()以及 Ajax 回调函数 callbackFunc()，关键代码如下：

```
var request = false;                                    //用于保存 XMLHttpRequest 对象的全局变量
var offsetE;                                            //定义一个全局变量，用于保存偏移量的对象
function getData(element,id){
        offsetE=element;                                //将当前的偏移量对象保存到全局变量中
        var url="getTip.jsp";                           //服务器地址
        var param ="id="+id+"&nocache="+new Date().getTime()    //请求参数
        request=httpRequest("get",url,true,callbackFunc,param); //调用请求方法
}
//Ajax 回调函数
function callbackFunc(){
 if(request.readyState == 4){
        if(request.status == 200){
                //设置工具提示的左边距
                document.getElementById("toolTip").style.left=(offsetE.offsetLeft+150)+"px";
                //设置工具提示的上边距
                document.getElementById("toolTip").style.top=offsetE.offsetTop+"px";
                //将工具提示信息添加到页面中
                document.getElementById("goodsIntroduce").innerHTML=request.responseText;
                document.getElementById("toolTip").style.display="block";   //显示工具提示
                request = false;
        }
}
}
```

（4）编写隐藏工具提示的自定义 JavaScript 函数 clearTip()，具体代码如下：

```
function clearTip(){
        document.getElementById("toolTip").style.display="none";          //隐藏工具提示
}
```

（5）创建 getTip.jsp 页，在该页中通过指定商品 ID 获取商品的详细信息，关键代码如下：

```
<%@ page contentType="text/html; charset=GBK" language="java" import="java.sql.*" %>
<%@ page import="com.core.ConnDB"%>
<%
        String id=request.getParameter("id");
        ConnDB conn =new ConnDB();
        String sql="select introduce from tb_goods where id="+id;
        ResultSet rs=conn.executeQuery(sql);
        if(rs.next()){
                out.print(rs.getString(1));
        }
%>
```

（6）在 index.jsp 页显示商品信息的单元格中添加 onMouseOver 事件和 onMouseOut 事件，在 onMouseOver 事件中调用 getData()方法显示工具提示信息，在 onMouseOut 事件中调用 clearTip()方法隐藏工具提示信息，关键代码如下：

```
<td align="center" onMouseOver="getData(this,<%=new_ID%>)" onMouseOut="clearTip()">
```

秘笈心法

根据本实例，读者可以实现在论坛中显示帖子信息时，快速浏览显示发帖人的详细介绍；在显示相片缩略

图时，快速浏览显示相片的详细介绍。

实例 252　动态多级联下拉列表

光盘位置：光盘\MR\10\252

高级

实用指数：★★★★

实例说明

多级联下拉列表是指一组相互关联的下拉列表，相邻的两个下拉列表是父子关系，改变父下拉列表的值，子下拉列表的值也随之改变。运行本实例，如图 10.14 所示，在页面中将显示一个三级联动下拉列表，在省和直辖市的下拉列表中选择省份，在地级市下拉列表中将显示出该省份的地级市信息，在地级市下拉列表中选择其中一个市，在县、县级市或区下拉列表中将显示区信息。

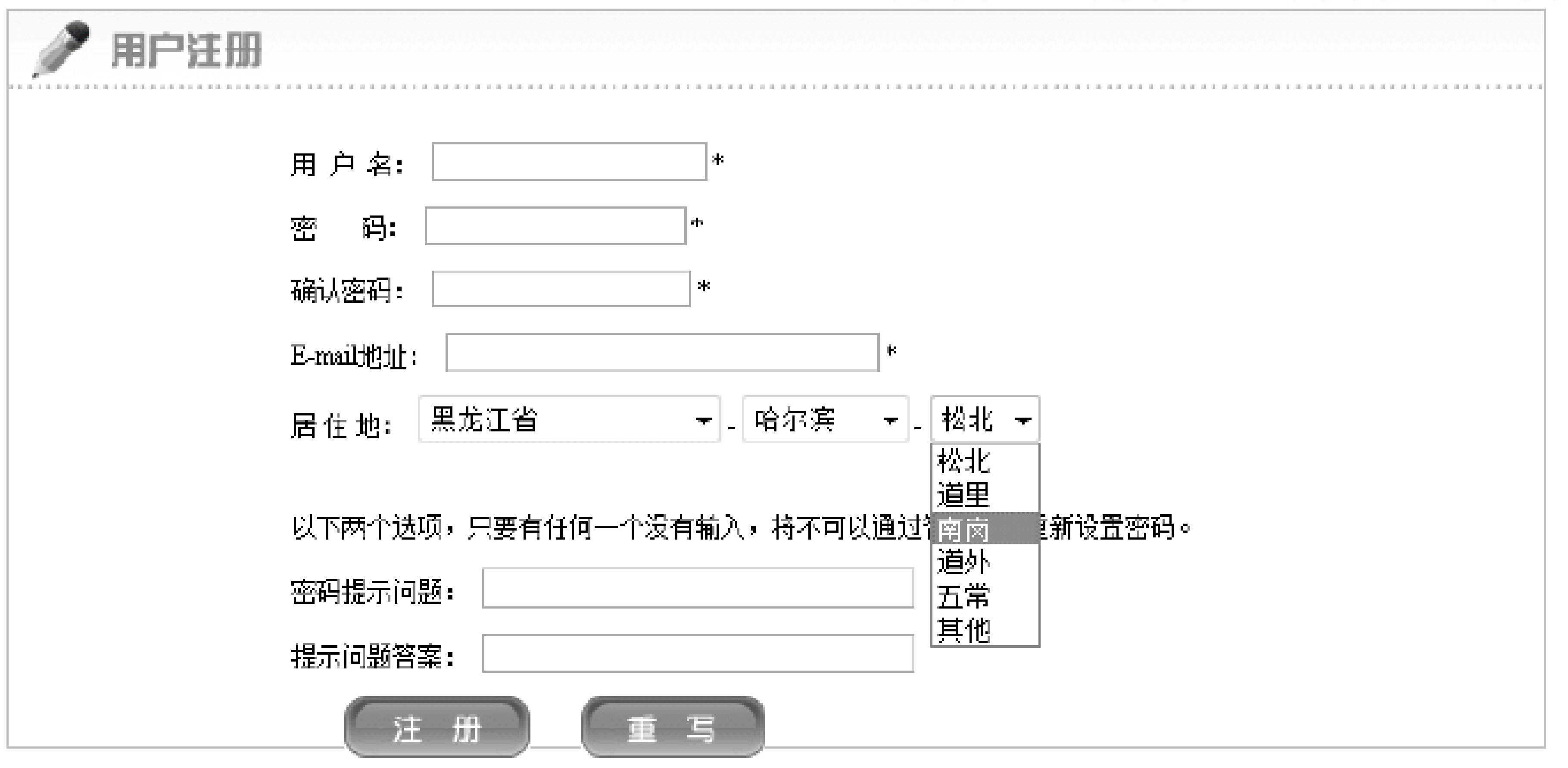


图 10.14　多级联下拉列表

关键技术

在实现本实例时，首先需要设计一个保存省市信息的 XML 文件，在该文件中，需要充分体现出省/直辖市、地级市和县/县级市/区之间的关系。接下来需要从 XML 文件中获取所需的信息，也就是解析 XML 文件。本实例中应用的是 dom4j 组件解析 XML 文件，下面将对应用 dom4j 解析 XML 文件进行介绍。

1. 构建 XML 文档对象

在解析 XML 文档前，需要构建要解析的 XML 文件所对应的 XML 文档对象。在获取 XML 文档对象时，首先需要创建 SAXReader 对象，然后调用该对象的 read()方法获取对应的 XML 文档对象。SAXReader 对象的 read()方法的原型如下：

```
public Document read(File file) throws DocumentException
```

参数说明

file：用于指定要解析的 XML 文件。

例如，获取 XML 文件 zone.xml 对应的 XML 文档对象的代码如下：

```
String fileURL = request.getRealPath("/xml/zone.xml");
SAXReader reader = new SAXReader();                    //实例化 SAXReader 对象
Document document = reader.read(new File(fileURL));    //获取 XML 文件对应的 XML 文档对象
```

2. 获取根节点

在构建 XML 文档对象后，就可以通过该 XML 文档对象获取根节点。dom4j 组件的 Document 对象的 getRootElement()方法可以返回指定 XML 文档的根节点。getRootElement()方法的原型如下：

```
public Element getRootElement()
```

返回值：Element 对象。

例如，获取 XML 文档对象 document 的根节点的代码如下：

```
Element country= document.getRootElement();            //获取根节点
```

3. 获取子节点

在获取根节点后，还可以获取其子节点，这可以通过 Element 对象的 element()或 elements()方法实现。下面将分别介绍这两个方法。

（1）element()方法

element()方法用于获取指定名称的第一个节点。该方法通常用于获取根节点中节点名唯一的一个子节点。element()方法的原型如下：

```
public Element element(String name)
```

参数说明

name：用于指定要获取的节点名。

返回值：Element 对象。

（2）elements()方法

elements()方法用于获取指定名称的全部节点。该方法通常用于获取根节点中多个并列的具有相同名称的子节点。elements()方法的原型如下：

```
public List elements(String name)
```

参数说明

name：用于指定要获取的节点名。

返回值：List 集合。

例如，要获取本实例中的 XML 文件 zone.xml 的 province 节点，可以应用 elements()方法，具体代码如下：

```
Element country = document.getRootElement();                           //获取根节点
List<Element> provinceList = country.elements("province");             //获取表示省份和直辖市的节点
```

4. 查询节点

在 dom4j 组件中，查询节点可以应用 Element 对象的 selectSingleNode()方法实现。Element 对象的 selectSingleNode()方法用于获取符合指定条件的唯一节点，该方法的原型如下：

```
public Node selectSingleNode(String xpathExpression)
```

参数说明

xpathExpression：XPath 表达式。XPath 表达式使用反斜杠“/”隔开节点树中的父子节点，从而构成代表节点位置的路径。如果 XPath 表达式以反斜杠“/”开头，则表示使用的是绝对路径，否则表示使用的是相对路径。如果使用属性，那么必须在属性名前加上@符号。另外，在 XPath 表达式中，也可以使用谓词。

例如，下面的表达式将返回 name 属性值等于“北京市”的 province 节点。

```
/country/province[@name='北京市']
```

例如，应用 selectSingleNode()方法获取 XML 文档的根节点 country 的 name 属性为“北京市”的子节点 province 的代码如下：

```
Element item = (Element) country.selectSingleNode("/country/province[@name='北京市']");
```

5. 获取属性值

使用 Element 对象的 attributeValue()方法可以获取指定节点的指定属性值，该方法的原型如下：

```
public String attributeValue(String name)
```

参数说明

name：用于指定要获取其值的属性名。

例如，要获取 province 节点的 name 属性值，可以使用下面的语句：

```
Element provinceElement.attributeValue("name")
```

■ 设计过程

（1）创建一个 XML 文件，名称为 zone.xml，用于保存省市信息。zone.xml 文件的关键代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<country name="中国">
<province id="00000" name="北京市">
```

```
        <city id="00001" name="北京" area="东城区,西城区,崇文区,宣武区,朝阳区,丰台区,
                石景山区,海淀区,门头沟区,房山区,通州区,顺义区,昌平区,大兴区,怀柔区,平谷区,
                密云区,延庆县">
        </city>
</province>
<province id="05000" name="吉林省">
        <city id="05001" name="长春" area="双阳区,德惠市,九台市,农安县,榆树市,南关区,
                宽城区,朝阳区,二道区,绿园区,经济技术开发区,高新区">
        </city>
        <city id="05002" name="延边朝鲜族自治州" area="延吉市,图们市,敦化市,珲春市">
        </city>
        <city id="05003" name="吉林" area="船营区,冒邑区,龙潭区,丰满区,蛟河市,桦甸市">
        </city>
        <city id="05004" name="白山" area="八道江区,江源区,临江区,抚松县,靖宇县">
        </city>
        <city id="05005" name="白城" area="洮北区,洮南区,大安市,镇赉县,通榆县,其他">
        </city>
        <city id="05006" name="四平" area="梨树县,伊通满族自治县,公主岭市,双辽市">
        </city>
        <city id="05007" name="松原" area="宁江区,长岭县,乾安县,扶余县,前郭">
        </city>
        <city id="05008" name="辽源" area="龙山区,西安区,东丰县,东辽县,其他">
        </city>
        <city id="05009" name="通化" area="东冒区,二道江区,梅河口市,集安市,通化县">
        </city>
</province>
……                          <!-- 省略了其他省市的节点信息 -->
</country>
```

（2）编写 index.jsp 文件，应用 DIV+CSS 进行布局，并在该文件的适当位置添加省/直辖市下拉列表、地级市下拉列表和县/县级市/区下拉列表，关键代码如下：

```
<select name="province" id="province">
 </select>
  -
  <select name="city" id="city">
  </select>
  -
  <select name="area" id="area">
  </select>
```

（3）创建 request.js 文件，用于封装 Ajax 请求服务器的方法。

（4）在 index.jsp 页面的<script>标签中首先导入 request.js 文件，然后在<script>标签内声明 3 个全局变量，分别是获取省份的 XMLHttpRequest 对象的全局变量、获取地级市的 XMLHttpRequest 对象的全局变量和获取县、区的 XMLHttpRequest 对象的全局变量。编写实例化 Ajax 对象的方法 getProvince()，用于向服务器发送请求，获取省份和直辖市，关键代码如下：

```
<script language="javascript" src="js/request.js"></script>
<script language="javascript">
var provinceRequest = false;                                  //声明获取省份的 XMLHttpRequest 对象的全局变量
var cityRequest = false;                                      //声明获取地级市的 XMLHttpRequest 对象的全局变量
var areaRequest = false;                                      //声明获取县、区的 XMLHttpRequest 对象的全局变量
//获取省份和直辖市
function getProvince(){
        var url="ZoneServlet";                                //服务器地址
        var param ="action=getProvince&nocache="+new Date().getTime(); //请求参数
        request=httpRequest("post",url,true,callback_getProvince,param);   //调用请求方法
}
```

（5）创建用于处理请求的 Servlet 实现类 ZoneServlet，在该 Servlet 的 doPost()方法中根据获取的 action 参数值来调用不同的方法。doPost()方法的具体代码如下：

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException{
      String action = request.getParameter("action");         //获取请求中的 action 参数值
      if ("getProvince".equals(action))
         getProvince(request, response);                      //获取省份的方法
      else if ("getCity".equals(action))
```

```
        getCity(request, response);                         //获取地区市的方法
    else if ("getArea".equals(action))
        getArea(request, response);                         //获取县区的方法
}
```

（6）在 ZoneServlet 中编写用于获取省份和直辖市的 getProvince()方法。首先获取保存市县信息的 XML 文件的完整路径，然后判断该 XML 文件是否存在，如果存在，则通过 dom4j 组件解析该文件，从中获取出省份和直辖市并连接为以逗号分隔的字符串，最后设置应答的类型为 HTML，并且输出由县和直辖市信息组成的字符串，如果没有获取到相关内容，则输出空的字符串。getProvince()方法的具体代码如下：

```
public void getProvince(HttpServletRequest request,HttpServletResponse response)
        throws ServletException, IOException {
        response.setCharacterEncoding("UTF-8");             //设置响应的编码方式
        String fileURL = request..getRealPath("/xml/zone.xml");  //获取 XML 文件的路径
        File file = new File(fileURL);
        Document document = null;                           //声明 Document 对象
        Element country = null;                             //声明根节点的 Element
        String result = "";
        if (file.exists()) {                                //如果文件存在，则读取该文件
                SAXReader reader = new SAXReader();         //实例化 SAXReader 对象
                try {
                        document = reader.read(new File(fileURL));  //获取 XML 文件对应的 XML 文档对象
                        country = document.getRootElement();        //获取根节点
                        //获取表示省份和直辖市的节点
                        List<Element> provinceList = country.elements("province");
                        Element provinceElement = null;
                        //将获取的省份连接为一个以逗号分隔的字符串
                        for (int i = 0; i < provinceList.size(); i++) {
                                provinceElement = provinceList.get(i);
                                result = result + provinceElement.attributeValue("name")+ ",";
                        }
                        result = result.substring(0, result.length() - 1);  //去除最后一个逗号
                } catch (DocumentException e) {
                        e.printStackTrace();
                }
        }
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.print(result);                                  //输出获取的市县字符串
        out.flush();
        out.close();
}
```

（7）在 index.jsp 页中编写获取省份直辖市的 Ajax 回调函数。首先需要将获取的省份名称字符串分隔为数组，然后通过循环将数组中的省份名称添加到下拉列表中，并且当下拉列表的第一个选项不为空时，还需要调用获取地级市的方法，获取默认的地级市信息，关键代码如下：

```
function callback_getProvince(){
 if(request.readyState == 4){
        if(request.status == 200){
                provinceArr=request.responseText.split(",");    //将获取的省份名称字符串分隔为数组
                for(i=0;i<provinceArr.length;i++){              //通过循环将省份名称添加到下拉列表中
                        document.getElementById("province").options[i]=new Option(provinceArr[i],provinceArr[i]);
                }
                if(provinceArr[0]!=""){
                        getCity(provinceArr[0]);                //获取地级市
                }
                request = false;
        }
 }
}
```

（8）为了让页面载入后即可获取到省份和直辖市信息，还需要在窗口的 onload 事件中调用 getProvince()函数，关键代码如下：

```
window.onload=function(){
        getProvince();                                      //获取省份和直辖市
}
```

（9）在 index.jsp 页的<script>中，编写实例化 Ajax 对象的方法 getCity()，用于向服务器发送请求，获取地级市，关键代码如下：

```
function getCity(selProvince){
 var url="ZoneServlet";                                                  //服务器地址
 var param ="action=getCity&parProvince="+selProvince+"&nocache="+new Date().getTime();
 request=httpRequest("post",url,true,callback_getCity,param);             //调用请求方法
}
```

（10）在 ZoneServlet 类中，编写用于获取省份或直辖市所对应的地区市信息的 getCity()方法。在该方法中通过 dom4j 组件解析 XML 文件，从中获取出指定省份或直辖市所对应的地级市信息并连接为以逗号分隔的字符串，最后设置应答的类型为 HTML，并且输出由地级市信息组成的字符串，如果没有获取到相关内容，则输出空的字符串。getCity()方法的具体代码如下：

```
public void getCity(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setCharacterEncoding("UTF-8");                          //设置响应的编码方式
        String fileURL = request.getRealPath("/xml/zone.xml");           //获取 XML 文件的路径
        File file = new File(fileURL);
        Document document = null;                                        //声明 Document 对象
        String result = "";
        if (file.exists()) {                                             //如果文件存在，则读取该文件
                SAXReader reader = new SAXReader();                      //实例化 SAXReader 对象
                try {
                        document = reader.read(new File(fileURL));       //获取 XML 文件对应的 XML 文档对象
                        Element country = document.getRootElement();     //获取根节点
                        String selProvince = request.getParameter("parProvince");   //获取的省份
                        selProvince = java.net.URLDecoder.decode(selProvince,"UTF-8");
                        Element item = (Element) country.selectSingleNode("/country/province[@name='"
                                        + selProvince + "']");           //获取指定 name 属性的省份节点
                        List<Element> cityList = item.elements("city");  //获取表示地级市的节点集合
                        Element cityElement = null;
                        for (int i = 0; i < cityList.size(); i++) {
                                cityElement = cityList.get(i);
                                result = result + cityElement.attributeValue("name") + ",";
                        }
                        result = result.substring(0, result.length() - 1);   //去除最后一个逗号
                } catch (DocumentException e) {
                        e.printStackTrace();
                }
        }
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.print(result);                                               //输出获取的地级市字符串
        out.flush();
        out.close();
}
```

（11）在 index.jsp 页面中，编写获取地级市信息的 Ajax 回调函数 callback_getCity()。在该函数中，首先将获取的市县名称字符串分隔为数组，然后清空地级市下拉列表（防止下拉列表框的内容重复添加），再通过循环将数组中的市县名称添加到下拉列表中，最后当下拉列表的第一个选项不为空时，还需要调用获取县/县级市/区的方法，获取默认的县、县级市和区信息。callback_getCity()函数的具体代码如下：

```
function callback_getCity(){
 if(request.readyState == 4){
        if(request.status == 200){
                cityArr=request.responseText.split(",");                 //将获取的市县名称字符串分隔为数组
                document.getElementById("city").length=0;                //清空下拉列表
                for(i=0;i<cityArr.length;i++){                           //循环将地级市名称添加到下拉列表中
                        document.getElementById("city").options[i]=new Option(cityArr[i],cityArr[i]);
                }
                if(cityArr[0]!=""){
                        getArea(document.getElementById("province").value,cityArr[0]);
                }
                request = false;
        }
 }
}
```

（12）在县/直辖市下拉列表的 onChange 事件中，调用 getCity()方法，获取地级市信息，关键代码如下：

```
<select name="province" id="province" onChange="getCity(this.value)"></select>
```

（13）在 index.jsp 页面中，编写实例化 Ajax 对象的方法 getArea()，用于向服务器发送请求，获取县、县级市和区信息，关键代码如下：

```
function getArea(selProvince,selCity){
 var url="ZoneServlet";//服务器地址
 var param ="action=getArea&parProvince="+selProvince+"&parCity="+selCity+"&nocache="
                    +new Date().getTime();
 request=httpRequest("post",url,true,callback_getArea,param);               //调用请求方法
}
```

（14）在 ZoneServlet 类中，编写获取县、县级市或区的 getArea()方法，关键代码如下：

```
public void getArea(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setCharacterEncoding("UTF-8");                              //设置响应的编码方式
        String fileURL = request.getRealPath("/xml/zone.xml");               //获取 XML 文件的路径
        File file = new File(fileURL);
        Document document = null;                                            //声明 Document 对象
        String result = "";
        if (file.exists()) {                                                 //如果文件存在，则读取该文件
                SAXReader reader = new SAXReader();                          //实例化 SAXReader 对象
                try {
                        document = reader.read(new File(fileURL));           //获取 XML 文档对象
                        Element country = document.getRootElement();         //获取根节点
                        String selProvince = request.getParameter("parProvince");   //获取选择的省份
                        String selCity = request.getParameter("parCity");    //获取选择的地级市
                        selProvince = java.net.URLDecoder.decode(selProvince,"UTF-8");
                        selCity = java.net.URLDecoder.decode(selCity,"UTF-8");
                        Element item = (Element) country.selectSingleNode("/country/province[@name='"
                                + selProvince + "']");
                        List<Element> cityList = item.elements("city");      //获取表示地级市的节点集合
                        //获取指定的地级市节点
                        Element itemArea = (Element) item.selectSingleNode("city[@name='" + selCity + "']");
                        result = itemArea.attributeValue("area");            //获取县、县级市或区
                } catch (DocumentException e) {
                        e.printStackTrace();
                }
        }
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.print(result);                                                   //输出获取的县、县级市或区字符串
        out.flush();
        out.close();
}
```

（15）在 index.jsp 页面中，编写回调函数 callback_getArea()。在该函数中，首先将获取的县、县级市和区名称字符串分隔为数组，然后清空县/县级市/区下拉列表（防止下拉列表框的内容重复添加），最后通过循环将数组中的县、县级市和区名称添加到下拉列表中。callback_getArea()函数的具体代码如下：

```
function callback_getArea(){
 if(request.readyState == 4){
        if(request.status == 200){
                areaArr=request.responseText.split(",");             //将获取的市县名称字符串分隔为数组
                document.getElementById("area").length=0;           //清空下拉列表
                for(i=0;i<areaArr.length;i++){                      //循环将县、县级市和区名称添加到下拉列表中
                        document.getElementById("area").options[i]=new Option(areaArr[i],areaArr[i]);
                }
                request =false;
        }
 }
}
```

（16）在地级市下拉列表框的 onChange 事件中调用 getArea()方法，获取县、县级市和区信息，关键代码如下：

```
<select name="city" id="city"
        onChange="getArea(document.getElementById('province').value,this.value)">
</select>
```

秘笈心法

本实例在重新为地级市下拉列表和县/县级市/区下拉列表添加选项时，首先需要清空该下拉列表，否则下拉列表的选项值会出现错误。例如，清空地级市下拉列表可以使用下面的代码：

```
document.getElementById("city").length=0;          //清空下拉列表
```

清空县/县级市/区下拉列表可以使用下面的代码：

```
document.getElementById("area").length=0;          //清空下拉列表
```

第 2 篇

文件管理篇

⏭ 第 11 章　文件基本操作及文件上传下载

⏭ 第 12 章　文件的批量管理

第11章

文件基本操作及文件上传下载

- 文件的基本操作
- 无组件的文件上传
- 通过组件实现文件上传
- 文件下载

11.1　文件的基本操作

无论开发 Web 应用程序还是桌面应用程序，在实现一些功能模块时，对文件的操作是不可避免的。所以，在学习文件上传技术之前，首先应该了解一下如何对文件进行基本操作，如文件的读取、写入、解压缩操作、遍历等。

实例 253　查看文件是否存在

光盘位置：光盘\MR\11\253

初级　实用指数：★★★★

实例说明

在对文件进行操作之前，首先应该确定文件是否存在，如果被操作的文件不存在，那么程序运行时将会出现异常。本实例将介绍如何判断指定位置的文件是否存在。运行本实例，在“输入文件路径”文本框中输入“D:\test.txt”，单击“查找”按钮，如果磁盘 D 目录下不存在 test.txt 文本文件，将弹出“文件不存在”的提示对话框，如图 11.1 所示。

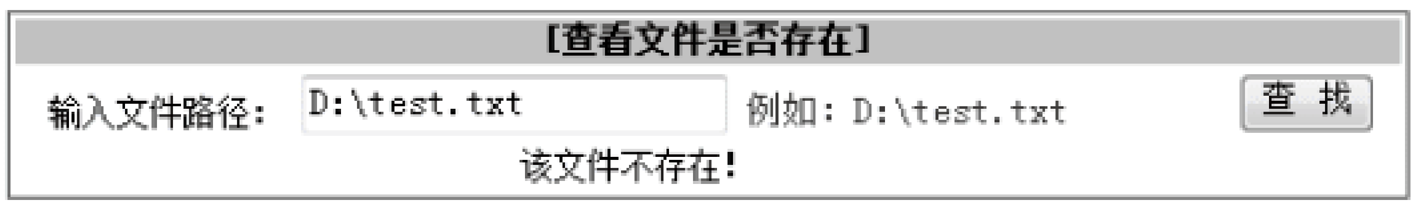


图 11.1　查看文件是否存在

关键技术

本实例主要应用 java.io.File 类的 exists()方法，该方法用于测试 File 对象表示的抽象路径名表示的文件或目录是否存在。在测试文件是否存在之前，首先应该创建一个表示文件和目录路径名的 File 对象，然后通过 File 对象的 exists()方法判断该文件是否存在，如果存在，exists()方法返回 true，否则返回 false，示例代码如下：

```
File file = new File("D:\test\test.txt");
boolean isHas = file.exists();
```

设计过程

（1）新建 Servlet 实现类 FileServlet，在该类的 doPost()方法中获取表单请求的文件路径信息，并判断该文件是否存在。doPost()方法的具体代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String filePath = request.getParameter("filePath");              //获取请求中的文件路径
        File file = new File(filePath);                                  //根据文件路径创建 File 对象
        boolean bool = file.exists();                                    //检查文件是否存在
        String str="";
        if(bool == true)
                str = "该文件存在！ ";
        else
                str = "该文件不存在！ ";
        request.setAttribute("Msg", str);                                //向 request 对象中添加信息
        request.getRequestDispatcher("index.jsp").forward(request, response);//请求转发
}
```

（2）在 index.jsp 页中将判断后的结果输出，关键代码如下：

```
<tr>
        <td    align="center"> </td>
        <%if(request.getAttribute("Msg") != null) {%>
        <td align="center"><%out.print(request.getAttribute("Msg"));} %></td>
        <td align="left" valign="top"> </td>
</tr>
```

秘笈心法

File 对象既可以表示文件，也可以表示目录，在程序中一个 File 对象可以代表一个文件或目录。它可用来对文件或目录及其属性进行基本操作。

实例 254	重命名文件 光盘位置：光盘\MR\11\254	初级 实用指数：★★★☆

实例说明

在应用程序对文件进行操作时，有时可能会将文件名称填写错误，此时需要重命名文件。本实例将介绍如何实现对文件的重命名操作。运行本实例，如图 11.2 所示，在“源文件名”文本框中输入文件名称和文件位置，在“新文件名”文本框中输入新的文件名称（直接输入文件名即可，不包含路径），单击“提交”按钮，即可实现对指定文件的重命名。



图 11.2　重命名文件

关键技术

本实例主要应用 File 类的 renameTo()方法实现文件的重命名，该语法结构如下：

```
public boolean renameTo(File newFile)
```

参数 newFile 是一个 File 对象，用于指定文件的新抽象路径名称。当重命名成功时，该方法返回值为 true，否则返回 false。

设计过程

（1）编写名为 RenameBean 的 JavaBean 类，在该类中实现对文件的重命名操作，关键代码如下：

```
public class RenameBean {
    private String fileName;                    //源文件名
    private String newFileName;                 //新文件名
    boolean fileInfo = false;
    public boolean renameFile() {
        File file = new File(fileName);         //根据源文件名创建 File 对象
        if (file.exists()) {                    //判断文件是否存在
            //根据源文件的父目录和新文件名，组合创建一个新的 File 对象
            File newFile = new File(file.getParent()+File.separator+newFileName);
            boolean res = file.renameTo(newFile);   //对源文件进行重命名
            if (res)
                fileInfo = true;                //重命名成功，返回 true
        }
        return fileInfo;
    }
    …//省略了属性的 getXXX()方法和 setXXX()方法
}
```

（2）在 index.jsp 页中，通过<jsp:useBean>动作导入 RenameBean 类，并通过<jsp:setProperty>动作将用户输入的源文件名和新文件名赋值给 RenameBean 对象中的属性，关键代码如下：

```
<jsp:useBean id="fileBean" class="bean.RenameBean" scope="page"/>
<jsp:setProperty name="fileBean" property="fileName" value="${param.fileName }"/>
<jsp:setProperty property="newFileName" name="fileBean" value="${param.newFileName }"/>
```

（3）在 index.jsp 页中输出重命名结果信息，关键代码如下：

```
<%
    if(fileBean.renameFile())
        out.println("重命名成功！");
    else
        out.println("重命名失败！");
%>
```

秘笈心法

在本实例中，根据源文件路径名来创建新的文件对象时，用到了 File 对象的 getParent()方法和 separator 静态属性。getParent()方法用于返回 File 对象表示的路径名中的父目录的路径字符串。例如，创建一个 File 对象路径为 E:\test\a.txt，调用 getParent()方法则返回 E:\test 字符串。separator 表示与系统有关的默认名称分隔符。在 UNIX 系统上，此字段的值为“/”；在 Microsoft Windows 系统上，它为“\\”。

实例 255　复制文件夹

光盘位置：光盘\MR\11\255

初级　实用指数：★★★★

实例说明

本实例将介绍如何完成对文件夹的复制操作，复制包括文件夹下的子文件夹以及文件。运行本实例，如图 11.3 所示，在“源文件夹”文本框中输入要复制的文件夹路径，在“目标文件夹”文本框中输入目标文件夹路径，单击“复制”按钮，程序将复制整个文件夹内容到目标文件夹中。

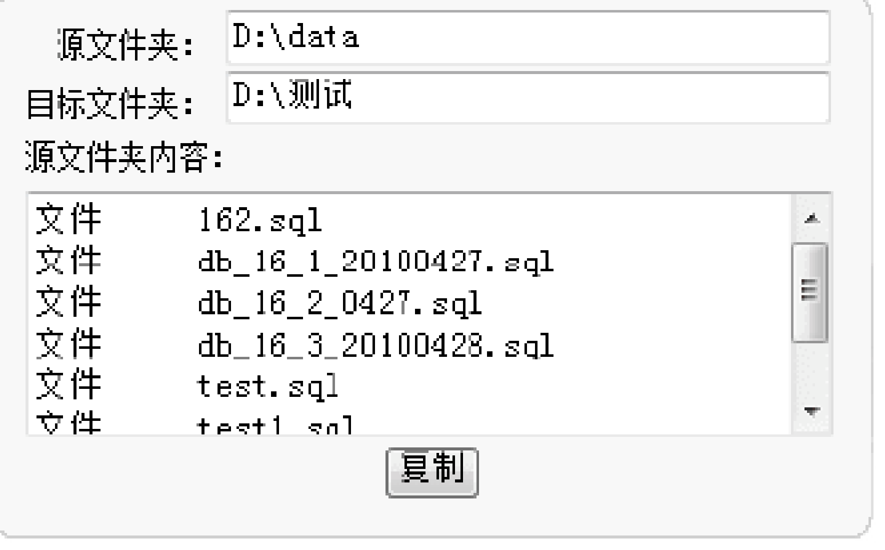


图 11.3　复制文件夹

关键技术

本实例主要应用 java.io.FileInputStream 类和 java.io.FileOutputStream 类来实现文件夹的复制。首先通过 File 对象中的 listFiles()方法获取文件夹路径下的所有文件对象组成的数组，然后循环文件对象数组，通过 FileInputStream 类读取文件内容，最后通过 FileOutoutStream 类将读出的文件内容写到目标文件夹中。下面对这两个类进行详细介绍。

1. 创建用于读取指定文件内容的数据流对象

FileInputStream 类用于从文件系统中的某个文件中获得输入字节，称为文件输入流。创建该类对象的语法结构如下：

```
FileInputStream fis = new FileInputStream(File file);
```

参数 file 为 File 类的实例对象，用于表示一个文件路径。

2. 创建文件输出流对象

FileOutputStream 类用于向文件中写入字节流数据，称为文件输出流。创建该类对象的语法结构如下：

```
FileOutputStream fos = new FileOutputStream(File file);
```

参数 file 为 File 类的实例对象，用于表示一个文件路径。

3. 读取文件字节流

读取文件字节流是通过 FileInputStream 对象的 read()方法，该方法每次只从文件中读取一个字节数据，如果读取的数据是文件的末尾，那么 read()方法则返回-1。

4. 向文件中写入字节流数据

通过 FileOutputStream 对象的 write()方法向文件中写入字节流数据。

设计过程

（1）创建 index.jsp 页，编写复制文件夹的 copy()方法，关键代码如下：

```
<%!
private void copy(File[] s,File d){                                    //定义复制文件夹的方法
        if(!d.exists())                                                //如果指定的目标目录不存在
                d.mkdir();                                             //创建目录
                for(int i=0;i<s.length;i++){                           //循环遍历源文件夹
                        if(s[i].isFile()){                             //如果为文件
                                try{

                                        FileInputStream fis=new FileInputStream(s[i]);    //根据文件路径名创建文件输入流
                                        File newFile = new File(d.getPath()+File.separator+s[i].getName()); //根据目标文件路径创建文件输出流

                                        FileOutputStream out=new FileOutputStream(newFile);
                                        int count;
                                        if((count=fis.read())>=0){     //从源文件中读取单个字节数据
                                                out.write(count);      //将字节数据写入目标文件中
                                        }
                                        out.close();                   //关闭流
                                        fis.close();
                                }catch(Exception e){
                                        e.printStackTrace();
                                }
                        }
                        if(s[i].isDirectory()){                        //如果为文件夹
                                File des=new File(d.getPath()+File.separator+s[i].getName());
                                des.mkdir();                           //创建目标文件目录
                                copy(s[i].listFiles(),des);            //继续调用 copy()方法
                        }
                }
}
%>
```

（2）从请求中获取源文件夹和目标文件夹信息，创建对应的 File 类对象，并且检测输入的文件夹信息是否存在，确定所有信息正确输入后，调用 copy()方法实现文件夹复制，关键代码如下：

```
<%
   File sourFolder=null,targetFolder=null;
   String source=request.getParameter("sourFolder");                   //获取用户输入的源文件夹地址
   String target=request.getParameter("targetFolder");                 //获取用户输入的目标文件夹地址
   if(source!=null){
        sourFolder=new File(source);                                   //创建源文件夹对象
       if(!sourFolder.isDirectory()||!sourFolder.exists()){            //如果该文件不存在
            out.print("<script language='javascript'>alert('源文件夹不存在')</script>");
       }
   }
   if(target!=null){
       targetFolder=new File(target);                                  //创建目标文件夹对象
        if(!targetFolder.exists()){                                    //如果目标文件不存在
            out.print("<script language='javascript'>alert('目标文件夹不存在')</script>");
        }else{
            copy(sourFolder.listFiles(),targetFolder);                 //调用复制文件夹的方法
        }
   }
%>
```

（3）在“源文件夹内容”列表框中显示源文件夹的内容，关键代码如下：

```
<%
    if(sourFolder!=null){
        File[] files=sourFolder.listFiles();
        for(int i=0;i<files.length;i++){
            out.println((files[i].isDirectory()?"文件夹\t":"文件\t")+ files[i].getName());
        }
    }
%>
```

秘笈心法

在本实例的复制文件夹的 copy()方法中，需要循环源文件夹的目录来获取源文件夹目录下的所有文件。这

些文件有可能是一个文件夹，需要通过 File 对象的 isDirectpry()方法来判断是否为文件夹，如果是文件夹则继续调用 copy()方法进行循环；如果是一个文件，则读取文件内容，并将其写入到目标文件中。

实例 256　**获取文件信息**　初级

光盘位置：光盘\MR\11\256　　实用指数：★★★★

实例说明

文件是存储在磁盘上的一组数据的集合，它在磁盘上有固定的存储格式和属性。其中属性包含文件名称、文件所在的路径、文件是否只读或隐藏以及文件大小等信息。本实例将介绍如何获取文件的相关信息。运行本实例，如图 11.4 所示，在“输入文件位置”文本框中输入完整的文件路径，单击“提交”按钮，将获取该文件的信息。



图 11.4　获取文件信息

关键技术

java.io.File 类的对象用于表示文件和目录路径名的抽象表示形式，File 类中提供了一些用于获取文件的相关属性信息的方法。在 File 类中包括的获取文件信息的常用方法及说明如表 11.1 所示。

表 11.1　用于获取文件属性信息的常用方法及说明

方　法	说　明
public String getName()	返回由抽象路径名表示的文件或目录的名称。该名称是路径名名称序列中的最后一个名称
public File getCanonicalFile() throws IOException	返回文件的全路径。如果发生 I/O 错误，则抛出 IOException 异常
public File getParentFile()	返回父目录的抽象路径名；如果没有指定父目录，则返回 null
public boolean isHidden()	判断 File 对象表示的文件是否是一个隐藏文件。返回 true 说明是隐藏文件
public boolean canWrite()	判断文件是否可被写入。返回 true 说明文件可以进行写入操作
public long lastModified()	返回文件的最后修改时间的 long 值，返回值以修改时间的毫秒值表示
public long length()	以字节为单位返回文件的大小

设计过程

（1）创建用于获取文件信息的 JavaBean 类 ShowFileInfoBean，关键代码如下：

```
public class ShowFileInfoBean {
private String filePath;                                    //文件路径字符串
```

```
Map<String, String> fileInfo = new HashMap<String, String>();      //定义存储文件对象的 Map 集合
public Map getFileInfo() {
        try {
                File file = new File(filePath);                    //创建文件对象
                if (!file.isFile())                                //判断是否为文件
                        return null;
                //以下代码表示将文件信息添加到 Map 集合中
                fileInfo.put("name", file.getName());
                fileInfo.put("absPath", file.getCanonicalPath());
                fileInfo.put("parentDir", file.getParentFile().getParent());
                fileInfo.put("hidden", file.isHidden()?"隐藏":"未隐藏");
                fileInfo.put("readOnly", file.canWrite()?"可写":"只读");
                fileInfo.put("lastModified", new Date(file.lastModified()).toLocaleString());
                fileInfo.put("length", String.format("%#,.2fK",file.length()/1024.0));
        } catch (IOException e) {
                e.printStackTrace();
        }
        return fileInfo;
}
…//此处省略了属性的 getXXX()方法和 setXXX()方法
}
```

（2）在 index.jsp 页中，通过<jsp:useBean>动作导入 ShowFileInfoBean，然后通过<jsp:setProperty>动作将用户输入的文件路径字符串赋值给 ShowFileInfoBean 对象中的属性，关键代码如下：

```
<jsp:useBean id="fileBean" class="bean.ShowFileInfoBean" scope="page"/>
<jsp:setProperty name="fileBean" property="filePath" value="${param.fileStr }"/>
```

（3）在 index.jsp 页中，获取 ShowFileInfoBean 对象中的 Map 集合属性，然后从 Map 集合中获取文件的信息，关键代码如下：

```
<table width="400" height="196" border="1" cellpadding="0" cellspacing="0"
      bordercolorlight="#AAAAAA" bordercolordark="#FFFFFF" >
   <tr>
     <td width="120" align="right">文件名称：</td>
     <td align="left">${fileBean.fileInfo['name'] } </td>
   </tr>
   <tr>
     <td align="right">文件路径：</td>
     <td align="left">${fileBean.fileInfo['absPath'] } </td>
   </tr>
   <tr>
     <td align="right">上级目录</td>
     <td align="left">${fileBean.fileInfo['parentDir'] } </td>
   </tr>
   <tr>
     <td align="right">只读属性：</td>
     <td align="left">${fileBean.fileInfo['readOnly']} </td>
   </tr>
   <tr>
     <td align="right">隐藏：</td>
     <td align="left">${fileBean.fileInfo['hidden']} </td>
   </tr>
   <tr>
     <td align="right">最后修改日期：</td>
     <td align="left">${fileBean.fileInfo['lastModified'] } </td>
   </tr>
   <tr>
     <td align="right">文件长度：</td>
     <td align="left">${fileBean.fileInfo['length'] } </td>
   </tr>
</table>
```

秘笈心法

本实例在 JavaBean 类中获取文件信息时，用到 HashMap 集合来保存文件的属性信息。HaspMap 类实现自 Map 接口，它允许任何类型的键和值对象，并允许将 null 用作键或值。HashMap 使用 put()方法存放键/值对象，使用 get()方法获取键的值。

实例 257　获取驱动器信息

光盘位置：光盘\MR\11\257

初级

实用指数：★★★★

实例说明

在 Web 应用程序的远程管理中，有时需要读取服务器的驱动器信息。本实例将介绍如何查询服务器上驱动器的分配情况。运行本实例，如图 11.5 所示，将显示当前服务器中驱动器的列表。

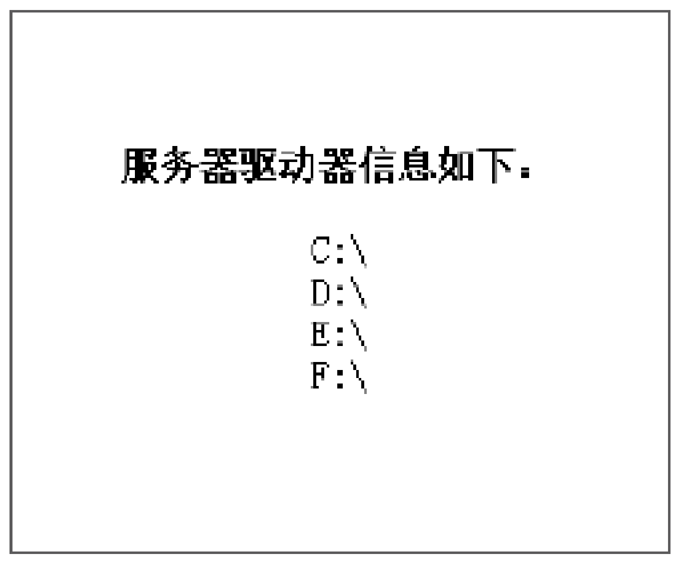


图 11.5　获取驱动器信息

关键技术

本实例主要应用 java.io.File 类的 listRoots()方法获取磁盘驱动器的分配情况，listRoots()方法返回一个 File 对象的数组，该数组中包含所有驱动器和映射的网络驱动器。listRoots()方法的语法结构如下：

```
public static File[] listRoots()
```

设计过程

创建获取驱动器信息的 index.jsp，在该页中获取驱动器信息并显示在表格中，关键代码如下：

```
<%
      File file[] = File.listRoots();
      for(int i=0;i<file.length;i++){
%>
        <tr>
             <td align="center"><%=file[i] %></td>
        </tr>
<%} %>
```

秘笈心法

通过 File 类的 listRoots()方法可以遍历“我的电脑”中的所有驱动器，包括网络驱动器的映射、移动硬盘、光驱和软驱等。

实例 258　读取属性文件

光盘位置：光盘\MR\11\258

初级

实用指数：★★★☆

实例说明

Java 提供了 Properties 类来操作存储信息的属性文件，属性文件以键/值对的方式存储信息。本实例将介绍如何从已经存在的属性文件中读取信息。运行本实例，如图 11.6 所示，从属性文件中读取连接数据库的配置信息。



属性文件键名	属性值	属性说明
DB_CLASS_NAME	com.mysql.jdbc.Driver	数据库驱动类
DB_URL	jdbc:mysql://localhost:3306/db_database10	连接URL
DB_USER	root	数据库用户名
DB_PWD	111	数据库密码

图 11.6　读取属性文件

关键技术

本实例在实现时主要应用了 java.io.FileInputStream 类和 java.util.Properties 类。FileInputStream 对象表示文件输入流对象，它是将文件内容以流的形式读取到内存中。Properties 类表示一个持久的属性集，它可保存在流中或从流中加载。属性列表中每个键及其对应值都是一个字符串，Properties 类的常用方法及说明如表 11.2 所示。

表 11.2　Properties 类的常用方法及说明

方　法	说　明
public String getProperty(String key)	用指定的键在属性文件的列表中搜索属性。如果在此属性列表中未找到该键，则接着递归检查默认属性列表及其默认值，如果未找到属性，则此方法返回 null
public void load(InputStream inStream)	从 InputStream 对象表示的输入流中读取属性列表（键和元素对）
public void load(Reader reader)	从 Reader 对象表示的输入字符流中读取属性列表（键和元素对）
Public Object setProperty(String key,String value)	设置属性列表的键和值
public Enumeration<?> propertyNames()	返回属性列表中所有键的枚举，如果在主属性列表中未找到同名的键，则包括默认属性列表中不同的键
public void loadFromXML(InputStream in)	将指定输入流中由 XML 文档所表示的所有属性加载到此属性表中
public long length()	以字节为单位返回文件的大小

设计过程

（1）创建名为 connDB.properties 的属性文件，在该文件中定义连接数据库的配置参数，关键代码如下：

```
DB_CLASS_NAME=com.mysql.jdbc.Driver
DB_URL=jdbc:mysql://localhost:3306/db_database10
DB_USER=root
DB_PWD=111
```

（2）新建 index.jsp 页，在该页中通过 FileInputStream 对象建立文件输入流，然后通过 Properties 对象的 load()方法读取属性文件，关键代码如下：

```
<%
        FileInputStream fis=new FileInputStream(application.getRealPath("connDB.properties"));
        Properties properties=new Properties();
        properties.load(fis);
%>
```

（3）通过 Properties 对象的 getProperty()方法获取指定的属性值并显示在页面中，关键代码如下：

```
<table width="500" height="214"   border="0" background="bg.gif" align="center">
        <tr>
          <td height="32" align="center">属性文件键名</td>
          <td align="center">属性值</td>
          <td align="center">属性说明</td>
        </tr>
        <tr>
          <td align="center">DB_CLASS_NAME</td>
          <td align="center"><%=properties.getProperty("DB_CLASS_NAME") %></td>
          <td align="center">数据库驱动类</td>
        </tr>
        <tr>
          <td align="center">DB_URL</td>
          <td align="center"><%=properties.getProperty("DB_URL") %></td>
          <td align="center">连接 URL</td>
        </tr>
        <tr>
          <td align="center">DB_USER</td>
          <td align="center"><%=properties.getProperty("DB_USER") %></td>
          <td align="center">数据库用户名</td>
        </tr>
```

```
        <tr>
            <td align="center">DB_PWD</td>
            <td align="center"><%=properties.getProperty("DB_PWD") %></td>
            <td align="center">数据库密码</td>
        </tr>
</table>
```

秘笈心法

在读取文件信息时，除了应用读取字节流的 java.io.FileInputStream 类以外，还可以应用读取字符流的 java.io.BufferedReader 类和 java.io.FileReader 类，这两个类都是 Reader 抽象类的子类。它们可以通过字符流的方式读取文本文件，并使用缓冲区，提高了读文本文件的效率。

实例 259　显示指定类型的文件

光盘位置：光盘\MR\11\259

高级　实用指数：★★★☆

实例说明

文件作为存储数据的单元，会根据数据类型产生很多分类，也就是所谓的文件类型。在对数据文件进行操作时，常常需要根据不同的文件类型来做不同的处理。本实例实现的是读取文件夹指定类型的文件并显示到表格控件中。这对项目开发中的文件分类起到了抛砖引玉的作用，读者可以对该实例进行相应的扩展，实例运行效果如图 11.7 所示。



图 11.7　显示指定类型的文件

关键技术

File 类位于 Java.io 类包中，它提供了多种对应文件和文件夹相关的操作，而本实例需要利用对文件夹相关的操作实现读取文件列表，并使用过滤器进行过滤。所以只能选用 File 类的 listFiles(FileFilter filter)方法，该方法的语法结构如下：

```
public File[] listFiles(FileFilter filter)
```

功能：返回抽象路径名数组，这些路径名表示此抽象路径名表示的目录中满足指定过滤器的文件和目录。

参数说明

filter：实现 FileFilter 接口的实例对象，该对象的 accept()方法用于实现文件的过滤。

设计过程

（1）创建 java.io.FileFilter 接口的实现类 CustomFilter，在该类中包含一个重写 FileFilter 接口的 accept()方法。在 accept()方法中，根据用户输入的扩展名来实现过滤所有不符合要求的文件。CustomFilter 类的具体代码如下：

```
public class CustomFilter implements FileFilter {
        private String extentName;                                  //用户设置的指定扩展名
        @Override
        public boolean accept(File pathname) {
                if (extentName == null || extentName.isEmpty())
                        return false;
                if (!extentName.startsWith("."))                    //判断扩展名前缀
                        extentName = "." + extentName;              //添加扩展名前缀
                extentName = extentName.toLowerCase();

                if (pathname.getName().toLowerCase().endsWith(extentName))   //判断扩展名与过滤文件名是否符合要求
                        return true;
                return false;
```

```
    }
    public String getExtentName() {
        return extentName;
    }
    public void setExtentName(String extentName) {
        this.extentName = extentName;
    }
}
```

（2）创建 index.jsp 页，在该页添加一个输入文件夹路径和文件扩展名的表单，然后将表单提交到本页。在本页中获取文件夹路径字符串和扩展名字符串，然后根据文件夹路径字符串创建一个 File 对象，调用该对象的 listFiles(FileFilter filter)方法，获取经过 CustomFilter 对象过滤后的文件对象数组，关键代码如下：

```
<%@ page import="com.lh.util.CustomFilter" %>
<%@ page import="java.io.*" %>
<%
    String filePath = request.getParameter("fileStr");          //获取文件夹路径字符串
    String extendStr = request.getParameter("extendStr");       //获取文件扩展名字符串
    File files[]=null;                                          //声明 File 对象数组
    if(filePath!=null&&extendStr!=null){
        CustomFilter fileFilter = new CustomFilter();           //创建 CustomFilter 对象
        fileFilter.setExtentName(extendStr);                    //为 CustomFilter 对象属性赋值
        File dir = new File(filePath);                          //创建文件路径的 File 对象
    if(dir.isDirectory()){                                      //判断是否为文件夹
        files = dir.listFiles(fileFilter);                      //获取经过过滤后的文件
    }
%>
```

（3）在 index.jsp 页中，循环过滤后的 File 对象数组，在表格中输出文件名称、文件大小以及最后修改时间，关键代码如下：

```
<table  border="1">
    <tr>
        <td>文件名称</td><td>文件大小</td><td>修改日期</td>
    </tr>
    <%
        if(files!=null){
            for(File file:files){
    %>
    <tr>
        <td><%=file.getName() %></td>
        <td><%=file.length() %></td>
        <td><%=new Date(file.lastModified()).toLocaleString() %></td>
    </tr>
        <% }
        }%>
</table>
```

秘笈心法

本实例主要是利用了 java.io.FileFilter 接口。首先自定义一个类实现自该接口，然后重写 accept()方法，最后在调用 File 对象的 listFiles(Filter filter)方法时，会自动执行实现自 FileFilter 接口的 accept()方法，从而实现最终目的。

实例 260	查找替换文本文件内容 光盘位置：光盘\MR\11\260	初级 实用指数：★★★☆

实例说明

文本替换几乎是所有文本编辑器都支持的功能，但是要限制在编辑器中才可以执行该功能。本实例实现了指定文本文件的内容替换，并且不需要在编辑器中打开文本文件，实例运行效果如图 11.8 和图 11.9 所示。读者可以将这个实例进行扩展，实现多文件内容的替换，这样就更能体现实用价值。



图 11.8　实例运行效果



图 11.9　替换前文本（左）与替换后文本（右）

关键技术

对文本文件的内容进行替换，离不开字符串的操作，而对于大量字符串的操作，StringBuilder 类是最快的。它可以动态改变字符串内容和不断向尾部追加新字符串。下面介绍本实例用到的 StringBuilder 类追加字符串的方法和 String 类替换字符串的方法。

（1）append()方法

StringBuilder 类的 append()方法可以向该类的对象尾部追加字符串文本，该方法的声明如下：

```
public StringBuilder append(String str)
```

该方法的作用是在字符串构建器的尾部追加参数指定的字符串。

（2）replace()方法

替换字符串要通过 String 类的 replace()方法实现，该方法的声明如下：

```
public String replace(CharSequence target, CharSequence replacement)
```

参数说明

❶ target：是要被替换的 char 值序列，即字符串。

❷ replacement：是替换的新字符串。

设计过程

（1）创建 ReplaceFile 类，在该类中包含一个用于替换文本内容的方法，关键代码如下：

```
public class ReplaceFile {
/***
 * 替换指定文件中的内容
 * @param filepath    文件路径
 * @param sourceStr  文件中包含的源内容
 * @param targetStr  替换后的内容
 * @return    替换成功返回 true，否则返回 false
 */
public static boolean replaceFileStr(String filepath,String sourceStr,String targetStr){
      try {
            FileReader fis = new FileReader(filepath);            //创建文件输入流
            BufferedReader br = new BufferedReader(fis);
            char[] data = new char[1024];                         //创建缓冲字符数组
            int rn = 0;
            StringBuilder sb=new StringBuilder();                 //创建字符串构建器
            while ((rn = fis.read(data)) > 0) {                   //读取文件内容到字符串构建器
                  String str=String.valueOf(data,0,rn);
                  sb.append(str);
            }
            fis.close();                                          //关闭输入流
            //从构建器中生成字符串，并替换搜索文本
            String str = sb.toString().replace(sourceStr, targetStr);
            FileWriter fout = new FileWriter(filepath);           //创建文件输出流
            fout.write(str.toCharArray());                        //把替换完成的字符串写入文件内
            fout.close();                                         //关闭输出流
            return true;
      } catch (FileNotFoundException e) {
            e.printStackTrace();
            return false;
      } catch (IOException e) {
```

```
            e.printStackTrace();
            return false;
        }
    }
}
```

（2）创建 index.jsp 页，在该页中包含一个选择文件的文本框和两个输入文本框的表单。表单信息直接提交到本页，在本页中获取表单元素的值，并调用 ReplaceFile 类中实现的方法替换文本文件中的内容，关键代码如下：

```
<%@ page import="java.io.*" %>
<%@ page import="java.net.*" %>
<%@ page import="com.lh.util.ReplaceFile" %>
<%
    String filePath = request.getParameter("pathStr");          //获取文件路径字符串
    String sourceStr = request.getParameter("sourceStr");       //获取文件扩展名字符串
    String targetStr = request.getParameter("targetStr");       //获取文件扩展名字符串
    boolean res =false;
    if(filePath!=null&&sourceStr!=null&&targetStr!=null){
        String path = new String(filePath.getBytes("ISO-8859-1"),"UTF-8");
        String source = new String(sourceStr.getBytes("ISO-8859-1"),"UTF-8");
        String target = new String(targetStr.getBytes("ISO-8859-1"),"UTF-8");
        res = ReplaceFile.replaceFileStr(path,source,target);    //调用替换文本的方法
    }
%>
```

秘笈心法

在应用程序开发过程中，经常需要使用 StringBuffer 来动态构造字符串，其中 append()是最常用的方法，因此建议读者掌握 StringBuffer 的应用。

实例 261 对文件夹创建、删除的操作

光盘位置：光盘\MR\11\261

初级

实用指数：★★★☆

实例说明

在 Web 应用开发过程中，在程序的后台有很多文件夹需要管理，经常需要添加新的文件夹来归类档案和网站资源，而且需要删除临时文件夹或无用的文件夹。运行本实例，如图 11.10 所示，输入文件夹的路径，单击“创建文件夹”或“删除文件夹”按钮，可以对文件夹进行创建和删除操作。

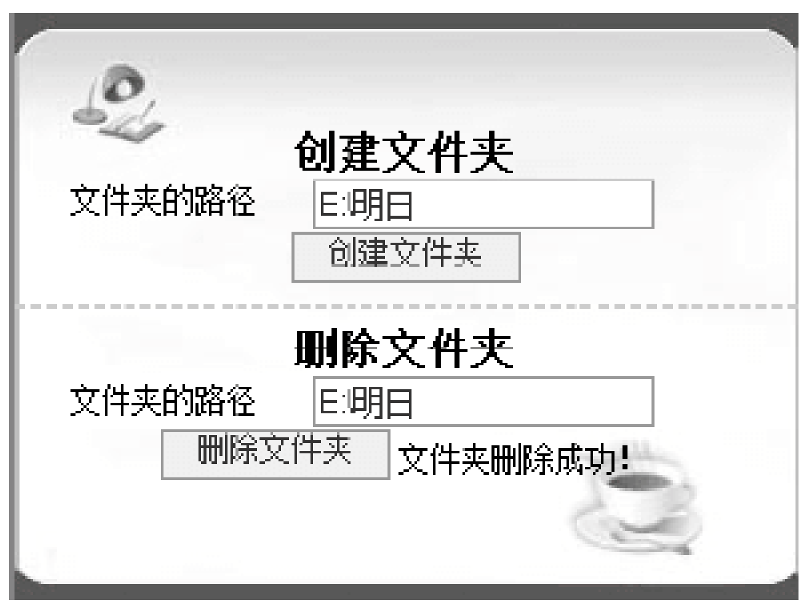


图 11.10 对文件夹创建、删除的操作

关键技术

本实例主要应用 java.io.File 类的 mkdir()方法和 delete()方法，它们都是非静态方法，需要创建 File 类的实例对象才能被调用，具体介绍如下。

（1）mkdir()方法

该方法的语法结构如下：

```
public boolean mkdir()
```

功能：创建 File 类表示的抽象路径名指定的目录，相当于创建文件夹目录。

（2）delete()方法

该方法的语法结构如下：

```
public boolean delete()
```

功能：删除此抽象路径名表示的文件或目录。如果此路径名表示一个目录，则该目录必须为空才能删除。

设计过程

（1）创建用于对文件夹创建和删除的工具类 FolderUtil，在该类中实现了两个主要方法，分别是创建文件夹的静态方法 createFolder()和删除文件夹的静态方法 deleteFolder()。FolderUtil 类的关键代码如下：

```
public class FolderUtil {
	/**
	 * 创建文件夹
	 * @param filepath 创建文件夹的路径
	 * @return 创建成功返回 true，否则返回 false
	 */
	public static boolean createFolder(String filepath){
		boolean res =false;				//声明一个布尔值变量，用于保存方法返回值
		File file = new File(filepath);			//根据路径字符串创建 File 对象
		res = file.mkdir();				//创建文件夹
		return res;
	}
	/**
	 * 删除文件夹
	 * @param filepath 删除文件夹的路径
	 * @return 删除成功返回 true，否则返回 false
	 */
	public static boolean deleteFolder(String filepath){
		boolean res = false;				//声明一个布尔值变量，用于保存方法返回值
		File file = new File(filepath);			//根据路径字符串创建 File 对象
		if(file.exists()&&file.isDirectory()){		//判断是否存在此路径
			res =file.delete();			//删除文件夹
		}
		return res;
	}
}
```

（2）在 index.jsp 页中获得表单的文件夹路径，然后判断单击的按钮是“创建文件夹”还是“删除文件夹”，调用 FolderUtil 类中的方法执行相应的操作，关键代码如下：

```
<%@ page import="com.lh.util.FolderUtil" %>
<%@ page import="java.io.*"%>
<%
	request.setCharacterEncoding("UTF-8");			//设置请求编码
	String filepath1 = request.getParameter("createDir");	//获取表单的创建文件夹路径
	String filepath2 = request.getParameter("deleteDir");	//获取表单的删除文件夹路径
	String submit = request.getParameter("Submit");		//获取提交按钮值
	boolean createRes = false;				//声明创建文件夹是否成功的变量
	boolean delRes = false;					//声明删除文件夹是否成功的变量
	if(filepath1!=null&&!filepath1.equals("")){		//判断是否为空
		if(submit!=null&&submit.equals("创建文件夹")){	//判断是否为“创建文件夹”的按钮
			createRes =FolderUtil.createFolder(filepath1);	//调用方法创建文件夹
		}
	}
	if(filepath2!=null&&!filepath2.equals("")){		//判断是否为空
		if(submit!=null&&submit.equals("删除文件夹")){	//判断是否为“删除文件夹”的按钮
			delRes =FolderUtil.deleteFolder(filepath2);	//调用方法删除文件夹
		}
	}
 %>
```

秘笈心法

当 File 对象表示的路径名为多级目录时，例如，D:\test\test1\test2\test3，那么使用 File 对象的 mkdir()方法则不能够创建这个多级目录结构。可以使用 File 对象中的另一个方法 mkdirs()，该方法表示创建抽象路径名指定的目录，包括所有必需但不存在的父目录。当且仅当已创建目录以及所有必需的父目录时，返回 true；否则返回 false。

实例 262 设置 Windows 的文件属性

光盘位置：光盘\MR\11\262

初级

实用指数：★★★☆

实例说明

文件是系统中保存数据的存储单元，它们可以设置不同的属性以保护关键文件，如把重要的配置文件设置为“只读”或“隐藏”属性、把系统调用的文件设置为“系统”属性，从而避免误操作删除需要使用的文件。本实例实现了对选定文件的属性设置，实例运行效果如图 11.11 所示。



图 11.11 设置 Windows 的文件属性

关键技术

本实例使用了 Runtime 类来执行系统命令，这是 Java 提供的一个运行环境交互的类，它可以执行系统命令。每个 Java 应用程序都有一个 Runtime 类的实例，它不能创建，只能通过该类的静态方法获取实例对象。该方法的声明如下：

```
public static Runtime getRuntime()
```

该方法返回与当前 Java 应用程序相关的运行时对象。Runtime 类的大多数方法是实例方法，并且必须根据当前的运行时对象对其进行调用。

该类的 exec()方法可以执行相应的系统命令，该方法的声明如下：

```
public Process exec(String command) throws IOException
```

该方法将返回一个 Process 类的实例对象，它代表本机进程，通过该实例的方法可以获取进程的输入流，从这个输入流中可以获取命令的返回结果。该方法的声明如下：

```
public abstract InputStream getInputStream()
```

使用该方法返回的输入流读取命令的返回结果，然后对该结果进行解析，从而获取指定文件的属性。

设计过程

（1）创建用于设置 Windows 文件属性的类 FileAttributeUtil，该类主要包含两个方法，分别是读取文件属性的方法和保存文件属性的方法，读取文件的属性信息主要是用于设置网页中复选框的选择状态。FileAttributeUtil 类的关键代码如下：

```
public class FileAttributeUtil {
    private boolean read = false;                              //只读属性
    private boolean hidden = false;                            //隐藏属性
    private boolean sys = false;                               //系统属性
    private boolean doc = false;                               //归档属性
    private String fileName="";                                //文件名
    private String filePath="";                                //文件路径
    /**
    * 读取文件属性
    * @param filepath 文件路径
    */
    public   void setFileAttribute(String filepath){
        File file = new File(filepath);                        //获取用户选择文件
        fileName = file.getName();
        filePath = file.getAbsolutePath();
        String command = "attrib " + file.getAbsolutePath();   //创建命令文本
        try {
            Process exec = Runtime.getRuntime().exec(command); //执行命令文本
            Scanner in = new Scanner(exec.getInputStream());   //创建命令执行环境的文本扫描器
            if (in.hasNextLine()) {
                String line = in.nextLine();                   //读取命令执行结果
                int of = line.indexOf(file.getPath());
                String attribStr = line.substring(0, of).trim();   //截取命令结果中文件的属性信息
```

```
                    doc = attribStr.contains("A");            //根据属性设置各复选框选中状态
                    hidden = attribStr.contains("H");
                    read = attribStr.contains("R");
                    sys = attribStr.contains("S");
                }
            } catch (IOException e1) {
                e1.printStackTrace();
            }
        }
        /**
        * 保存文件的属性设置
        * @param filepath 文件路径
        * @param str 文件属性的命令字符串
        * @return 保存成功返回 true，否则返回 false
        */
        public static boolean saveFileAttribute(String filepath,String str){
            File file = new File(filepath);                   //根据文件路径创建 File 对象
            String command = "attrib "+file.getPath()+str;    //设置文件属性的命令
            try {
                Runtime.getRuntime().exec(command);           //执行命令
                return true;
            } catch (IOException e1) {
                e1.printStackTrace();
                return false;
            }
        }
}
```

（2）新建 index.jsp 页，该页中主要包含一个选择文件的表单，在系统中选择文件后，将提交表单到本页并调用 FileAttributeUtil 类的 setFileAttribute()方法获取文件的属性信息，关键代码如下：

```
<%@ page import="com.lh.util.FileAttributeUtil" %>
<%
        request.setCharacterEncoding("UTF-8");               //设置请求编码格式
        String filePath = request.getParameter("pathStr");   //获取文件路径字符串
        FileAttributeUtil fileAttr = new FileAttributeUtil(); //创建用于操作文件属性的类
        if(filePath!=null&&!filePath.equals("")){           //判断字符串是否为空
            fileAttr.setFileAttribute(filePath);             //调用方法获取文件属性
        }
%>
```

（3）在 index.jsp 页中，根据获取的文件属性设置复选框是否为选中状态，关键代码如下：

```
<input type="checkbox" id="read" value="read"    <%if(fileAttr.isRead()){out.println("checked='checked'");} %> />只读
<input type="checkbox" id="hidden" value="hidden"    <%if(fileAttr.isHidden()){out.println("checked='checked'");} %> />隐藏
<input type="checkbox" id="doc" value="doc" <%if(fileAttr.isDoc()){out.println("checked='checked'");} %>/>归档
<input type="checkbox" id="sys" value="sys" <%if(fileAttr.isSys()){out.println("checked='checked'");} %>/>系统
```

（4）应用 JavaScript 方法，设置选中复选框的命令参数字符串，也就是设置文件属性的命令参数值。JavaScript 方法的关键代码如下：

```
function getCheckboxValue(){
        var readBox = document.getElementById("read");       //只读属性
        var hiddenBox = document.getElementById("hidden");   //隐藏属性
        var docBox = document.getElementById("doc");         //归档属性
        var sysBox = document.getElementById("sys");         //系统属性
        var str="";
        if(readBox.checked){
            str+=" +r";                                      //只读属性命令
        }else{
            str+=" -r";
        }
        if(hiddenBox.checked){
            str+=" +h";                                      //隐藏属性命令
        }else {
            str+=" -h";
        }
        if(docBox.checked){
            str+=" +a";                                      //归档属性命令
        }else{
            str+=" -a";
```

```
        }
        if(sysBox.checked){
                str+=" +s";                                 //系统属性命令
        }else {
                str+=" -s";
        }
        return str;
}
```

（5）设置完文件属性之后，单击“提交”按钮将表单信息提交到 save.jsp 页中，在该页中获取文件路径字符串和复选框的值，调用 FileAttributeUtil 类的 saveFileAttribute()方法保存文件属性设置。save.jsp 页的关键代码如下：

```
<%@ page import="com.lh.util.FileAttributeUtil" %>
<%
        request.setCharacterEncoding("UTF-8");                  //设置请求编码
        String str = request.getParameter("checkboxStr");       //获取复选框字符串
        String filePath = request.getParameter("pathStr");      //获取文件路径字符串
        boolean res = false;
        if(filePath!=null&&!filePath.equals("")){              //判断字符串是否为空
                res = FileAttributeUtil.saveFileAttribute(filePath,str);   //调用方法保存文件属性
        }
        if(res){%>                                             //如果保存方法返回值为 true
                <script type="text/javascript">
                        alert("文件属性修改成功！");            //弹出提示对话框
                        window.location.href="index.jsp";       //页面跳转到 index.jsp
                </script>
        <%}
%>
```

秘笈心法

Runtime 类的 exec()方法执行时，相当于在系统中应用命令行工具 cmd.exe 来执行命令。例如，在系统的 D 盘中有一个名为 test.txt 的文件，如果要对该文件加只读属性，在 cmd.exe 中的命令为“attrib D:\test.txt +r”，在程序中，Runtime 类的 exec()方法同样执行的是该命令。

实例 263　访问类路径上的资源文件

光盘位置：光盘\MR\11\263　　初级　　实用指数：★★★☆

实例说明

在实际开发中，经常需要访问一些资源文件，这些资源一般都存储在 Web 应用的指定类路径中。本实例将介绍如何从类路径中读取资源文件，程序的运行结果如图 11.12 所示。

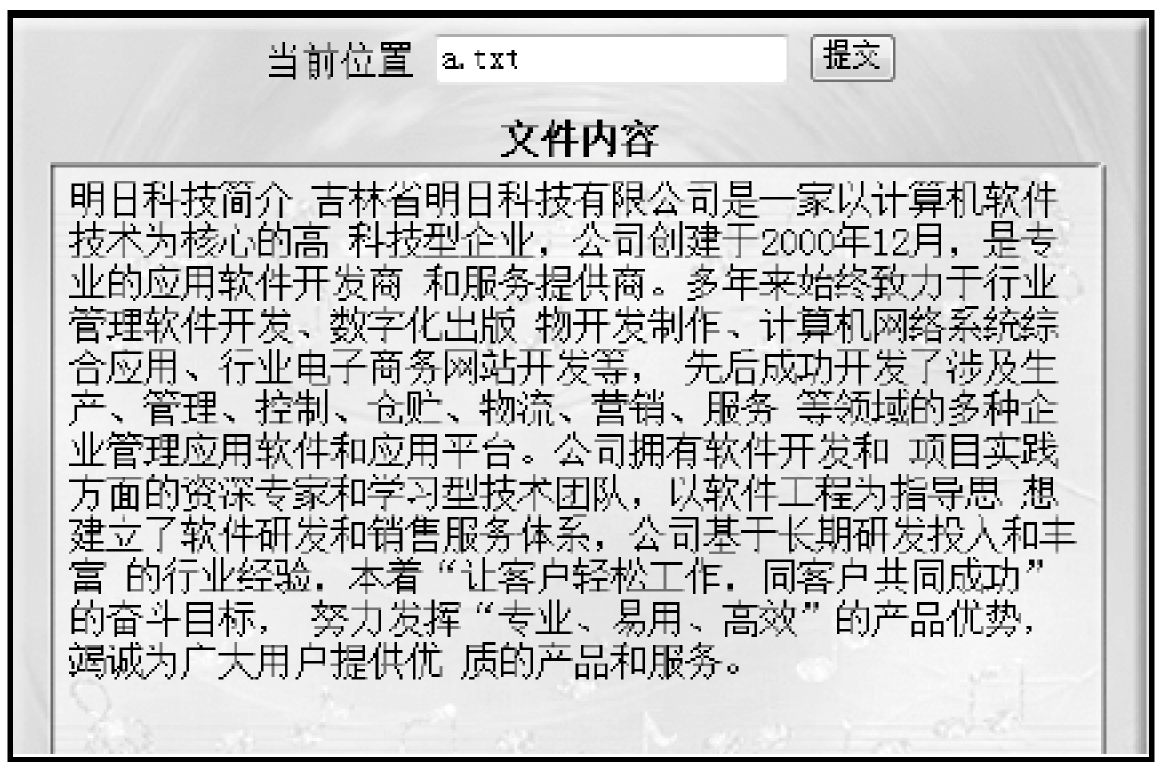


图 11.12　访问类路径上的资源文件

关键技术

本实例主要应用了 ServletContext 接口的 getRealPath(String path)方法，该方法根据参数 path 指定的虚拟路径，返回文件系统中的一个真实的路径。

设计过程

（1）创建 FileUtil 类，在该类中实现了一个用于读取文件内容的方法，关键代码如下：

```
public class FileUtil {
 public static String getFileTxt(String filePath){
        File file = new File(filePath);                         //创建 File 对象
        String fileTxt="";                                      //声明字符串对象，用于保存文件内容
```

```
        if (file.exists() && file.isFile()) {                       //判断是否为文件
            try {
                FileReader fr = new FileReader(file);               //创建字符输入流对象
                BufferedReader br = new BufferedReader(fr);         //从字符输入流中读取文本
                String line = br.readLine();                        //读取一行文本
                StringBuffer sb = new StringBuffer("");
                while(line!=null){                                  //如果当前行不为空
                    sb.append(line);                                //将读取出的文本添加到 StringBuffer 对象中
                    line = br.readLine();                           //读取下一行
                }
                br.close();                                         //关闭文件流
                fr.close();
                fileTxt = sb.toString();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }else{
            fileTxt = "文件"+file.getName()+"不存在！ ";
        }
        return fileTxt;
    }
}
```

（2）新建 index.jsp 页，在该页中调用 FileUtil 类的 getFileTxt()方法，获取文件的内容并赋值给变量 fileTxt，关键代码如下：

```
<%
    String fileTxt ="";
    request.setCharacterEncoding("UTF-8");                          //设置请求编码格式
    String filePath = request.getParameter("fileStr");              //获取用户输入的地址
    if (filePath != null) {                                         //判断是否为空
        String path = pageContext.getServletContext().getRealPath(filePath); //获取文件真实路径
        fileTxt = FileUtil.getFileTxt(path);                        //调用方法读取文本内容
    }
%>
```

（3）在 index.jsp 页的表格中输出变量 fileTxt 的值，关键代码如下：

```
<td><%=fileTxt %> </td>
```

秘笈心法

本实例在读取文本中应用到了 BufferedReader 类，该类用于从字符输入流中读取文本，缓冲各个字符，从而实现字符、数组和行的高效读取。

实例 264	实现永久计数器 光盘位置：光盘\MR\11\264	初级 实用指数：★★★★

实例说明

网站的计数器有很多种实现方式，如将计数值累加到系统变量、session、Servlet 上下文以及数据库和文件中。本实例以读写文件的方式实现永久性的计数器，无论是关闭浏览器还是重启服务器，即使关闭数据库都不会影响计数器的计数信息，实例的运行效果如图 11.13 所示。



图 11.13　实现永久计数器

关键技术

本实例首先通过 FileReader 类的 reader()方法读取计数值，并将数值加 1，再将计数值通过 FileWriter 类的 write()方法写入保存计数值的文件中。下面对这两个类进行具体介绍。

（1）FileReader 类

该类可以通过字符流的方式读取文本，并使用缓冲区提高读文本文件的效率。FileReader 类实现自 Reader

接口，其中包含一个 read(char[] cbuf)实现方法，该方法用于将字符读入数组，并返回读取的字符数，如果已到达流的末尾，则返回-1。read()方法的语法结构如下：

```
public int read(char[] cbuf) throws IOException
```

参数说明

cbuf：目标缓冲区。

（2）FileWriter 类

该类是字符输出流 Writer 抽象类下的子类。通过 FileWriter 可以以字符流的方式并通过缓冲区把数据写入文本文件，这也提高了写文本文件的效率。该类中包含一个实现 Writer 抽象类的 write()方法，该方法的语法结构如下：

```
public void write(String str) throws IOException
```

参数说明

str：要写入的字符串。

设计过程

（1）新建一个计数器文件 count.txt，为了方便读取，将该文件放到与 JSP 文件的同一目录下。

（2）创建 index.jsp 页，在该页中定义一个 long 型全局变量 count 作为计数器，关键代码如下：

```
<%!long count; %>
```

（3）在 index.jsp 页中，从文中读取上次访问的计数值，将其加 1 后再写入 count 文件中，关键代码如下：

```
<%
    File file=new File(getServletContext().getRealPath("count.txt"));        //创建文件对象
    FileReader reader=new FileReader(file);
    char[] cbuf=new char[(int)file.length()];                                  //创建字符数组
    reader.read(cbuf);                                                         //读取指定文件
    reader.close();                                                            //关闭流
    count=Long.valueOf(new String(cbuf)).longValue()+1;                        //将文件数据加 1
    FileWriter write=new FileWriter(file);
    write.write(new String(count+""));                                         //向文件中写数据
    write.close();                                                             //关闭流
%>
```

（4）在 index.jsp 页中显示全局变量 count 为当前网页访问量，关键代码如下：

```
<tr>您是本站第<%=count %>位访客</tr>
```

秘笈心法

为了更好地让访问者知道当前网站的访问量，在设计网页时可以更明显地突出网站的访问量的问题，这样可以更好地增加网站的访问量。

实例 265 从文本文件中读取注册服务条款

光盘位置：光盘\MR\11\265

初级

实用指数：★★★☆

实例说明

在网站中，如果用户想进行注册，在其要注册时，首先跳转的页面不会是直接填写注册信息的注册页面，而是需要先将拟定好的“服务条款”或“注册协议”显示给用户，让用户了解应该遵守的协议和能够享受到的服务，但是其内容有时是复杂且庞大的，如果直接写在网页中，对网站的维护会增加难度。为了解决这一问题，本实例实现了将其写到文件中，再由页面动态加载文件内容。如果要对文本内容进行修改，直接在文本文件中修改即可。运行效果如图 11.14 所示。

关键技术

实现本实例可以使用 FileReader 类的 read()方法，它能够以字符流的方式读取文件的文本信息。read()方法

的语法结构在实例 264 中已经介绍过，此处不再赘述。

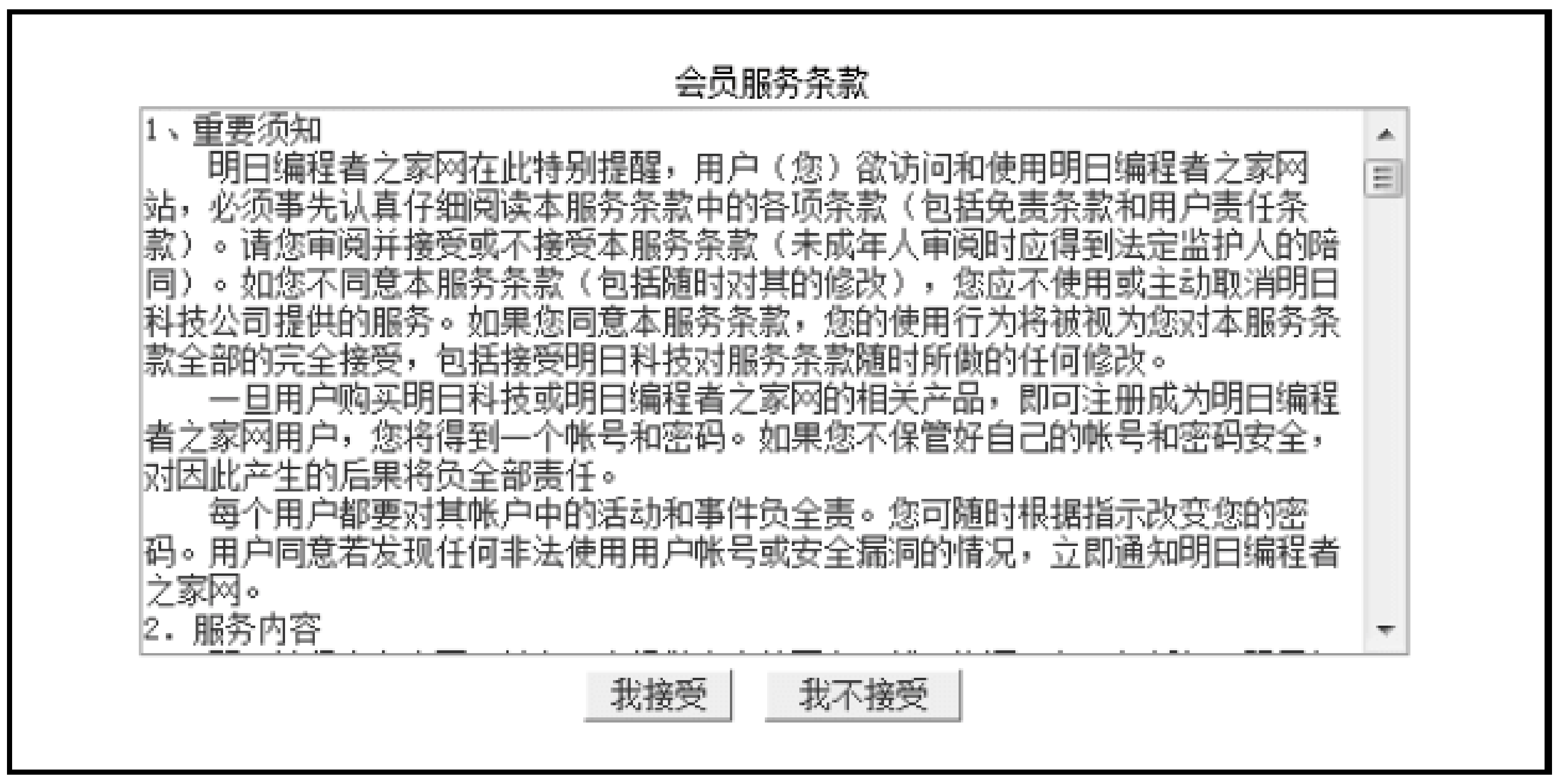


图 11.14　读取注册服务条款

设计过程

（1）创建一个 xieyi.txt 文件，其内容就是“服务条款”或“注册协议”的内容。

（2）创建 index.jsp 页，在该页面中定义读取文本信息所需的全局变量，关键代码如下：

```
<%!
    File file;                          //定义 File 对象
    FileReader reader;                  //定义 FileReader 对象
    char[] cbuf;                        //定义字符数组变量
%>
```

（3）实例化 FileReader 类，并设定数组的长度为文本文档的大小，最后调用 read()方法读取文本信息，关键代码如下：

```
<%
   file=new File(getServletContext().getRealPath("s.txt"));    //创建文件对象
   cbuf=new char[(int)file.length()];                          //创建字符数组
   reader=new FileReader(file);                                //创建 FileReader 对象
   reader.read(cbuf);                                          //从文件读取数据
%>
```

（4）将字符数组生成字符串显示到页面中，关键代码如下：

```
<textarea cols="75" rows="14"><%=String.valueOf(cbuf)%></textarea>
```

秘笈心法

根据本实例，读者还可以实现在开发在线论坛中的用户注册模块时，显示用户应遵守的网站协议。

实例 266　提取文本文件内容保存到数据库

初级

光盘位置：光盘\MR\11\266

实用指数：★★★★

实例说明

在网站的应用中，有时有很多的数据要大量地进行录入。例如，一些公司的员工信息，由于员工人数可能很多不方便进行手工录入，因此，需要使用原有的一些存储员工信息的文件，利用程序将这些文件信息导入到数据库中。本实例将实现从文件导出数据到数据库中的功能，运行本实例，如图 11.15 所示，单击“执行”按钮即可将指定的文件信息导入数据库中。文本文件的内容如图 11.16 所示。

关键技术

本实例在读取文件信息时，应用到了 java.io.FileReader 类，该类可以通过字符流的方式读取文件。FileReader 类的用法在实例 264 中已经介绍过，此处不再赘述。

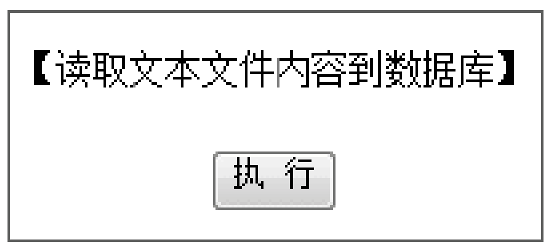


图 11.15　读取文件内容到数据库

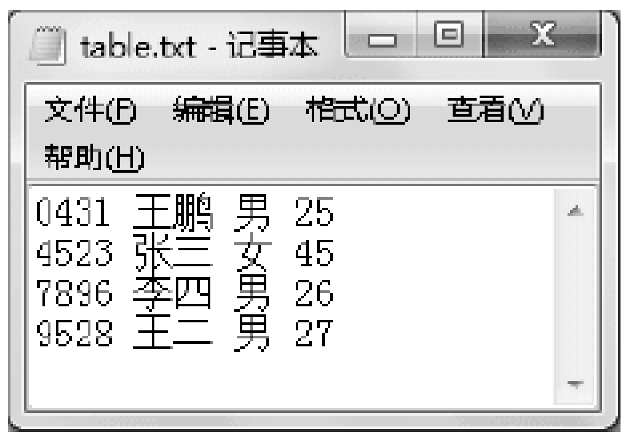


图 11.16　文本文件的内容

设计过程

（1）本实例应用的是 MySQL 数据库，在数据库 db_database11 中创建数据表 tb_emp，在该表中定义 id、name、sex 和 age 4 个 varchar 型的字段。

（2）创建一个包含员工信息的文本文件 table.txt，每一行为一个员工信息，而每个员工信息的字段用空格分隔。

（3）创建数据库连接类 DBCon，关键代码如下：

```
public class DBCon {
 private static Connection conn = null;
 public static Connection getConn(){
        try {
                Class.forName("com.mysql.jdbc.Driver");                    //加载数据库连接驱动
                String user="root";                                        //用户名
                String pwd="111";                                          //密码
                String url="jdbc:mysql://localhost:3306/db_database11";    //连接 URL
                conn=DriverManager.getConnection(url, user, pwd);          //获取连接
        } catch (Exception e) {
                e.printStackTrace();
        }
        return conn;
 }
}
```

（4）创建数据库操作类 EmpDao，在该类中主要包含一个读取文本文件内容保存到数据库的方法，该方法包含一个表示文件路径的 String 类型的参数，关键代码如下：

```
 public boolean readDataToMySQLDB(String filePath) throws SQLException{
        Connection conn = null;
        boolean res = false;
        try{
                conn = DBCon.getConn();                                    //创建数据库连接
                File file=new File(filePath);                              //创建 File 对象
                FileReader reader=new FileReader(file);                    //创建字符流对象
                char[] cbuf=new char[(int)file.length()];                  //根据文件长度创建字符数组
                reader.read(cbuf);                                         //读取文件信息
                reader.close();                                            //关闭流
                String[] table=new String(cbuf).split("\r\n");             //分割文本数据
                String[][] row=new String[table.length][];
                int num=0;
                String sql = "insert into tb_emp values(?,?,?,?)";         //SQL 语句
                PreparedStatement stmt=conn.prepareStatement(sql);         //创建预编译 SQL 语句对象
                for(int i=0;i<table.length;i++){                           //循环每一行
                        row[i]=table[i].split(" ");                        //对每一行进行分割
                        stmt.setString(1, row[i][0]);                      //每一行中的编号
                        stmt.setString(2, row[i][1]);                      //每一行中的姓名
                        stmt.setString(3, row[i][3]);                      //每一行中的年龄
                        stmt.setString(4, row[i][2]);                      //每一行中的性别
                        num=num+stmt.executeUpdate();                      //累加更新所影响的行数
                }
                if(num>0)
                        res = true;
        }catch(Exception ex){
                ex.printStackTrace();
        }finally{
                conn.close();
        }
```

```
        return res;
}
```

（5）创建 index.jsp 页，在该页中获取 table.txt 文件的路径，然后调用 EmpDao 类的方法，将文本文件内容保存到数据库，关键代码如下：

```
<%
        request.setCharacterEncoding("UTF-8");                    //设置请求编码
        String submit = request.getParameter("submit");           //获取按钮的值
        String filePath = application.getRealPath("table.txt");   //获取文件路径
        if(submit!=null){
                EmpDao.getInstance().readDataToMySQLDB(filePath);    //调用方法保存文件内容到数据库
        }
%>
```

秘笈心法

在读取文本内容时，也可以应用 BufferedReader 类结合 FileReader 类一起使用。它们都是 Reader 抽象类的子类，它们可以使用缓冲区，提高读文本文件的效率。

实例 267	将图片文件保存到数据库 光盘位置：光盘\MR\11\267	初级 实用指数：★★★★

实例说明

在向数据库保存数据的过程中，可能会遇到如图片、声音等字节文件。通常将这类文件存入到数据库中有两种方法，本实例演示第一种，即将图片转换成字节流插入到表格对应的列中，另一种将在实例 270 中介绍。利用这种方式可以实现对图片安全性的保护，实例运行效果如图 11.17 所示。



图 11.17　将图片保存到数据库

注意：保存图片的列的类型与使用的数据库有关，本实例使用的是 MySQL 数据库，列类型使用的是 LONGBLOB 类型，LONGBLOB 是一个二进制大对象，可容纳可变数量的数据。读者可以根据自己使用的数据库来决定列的类型。

关键技术

FileInputStream 可以从文件系统中的某个文件中获得输入字节流。它用于读入诸如图像数据之类的原始字节流。FileInputStream 类常用的方法如表 11.3 所示。

表 11.3　FileInputStream 类常用的方法

方　法　名	作　　用
FileInputStream(File file)	通过打开一个到实际文件的连接来创建一个 FileInputStream，该文件通过文件系统中的 File 对象 file 指定
public int available()	返回下一次对此输入流调用的方法可以不受阻塞地从此输入流读取（或跳过）的估计剩余字节数
public int read(byte[] b)	从此输入流中将最多 b.length 个字节的数据读入一个 byte 数组中
close()	关闭此文件输入流并释放与此流有关的所有系统资源

设计过程

（1）创建名为 PictureInfo 的 JavaBean 类，该类用于封装图片信息，关键代码如下：

```
public class PictureInfo {
        private int id;                                          //编号
        private String pictureName;                              //图片名称
        private String pictureType;                              //图片类型
        private File picturePath;                                //图片文件对象
        …//省略了属性的 getXXX()和 setXXX()方法
}
```

（2）创建数据库操作类 PictureDao，在该类中主要包含一个保存图片的方法 saveImage()，关键代码如下：

```
public boolean saveImage(PictureInfo picture) throws SQLException{
        Connection conn = null;
        boolean res = false;
        try{
                File pic = picture.getPicturePath();             //获取图片文件对象
                FileInputStream    fs = new FileInputStream(pic); //创建图片文件输入流
                byte[] b=new byte[(int)pic.length()];            //用来保存图片的字节数组
                fs.read(b);                                      //读取图片字节保存到字节数组
                conn = DBCon.getConn();                          //获取数据库连接
                String sql = "insert into tb_img(name,type,img) values(?,?,?)";
                PreparedStatement pstmt = conn.prepareStatement(sql);   //创建预编译对象
                pstmt.setString(1, picture.getPictureName());   //设置图片名称
                pstmt.setString(2, picture.getPictureType());   //设置图片类型
                pstmt.setBinaryStream(3, fs,(int)pic.length()); //设置图片原始字节流
                int i = pstmt.executeUpdate();                   //执行 SQL 语句
                if(i>0)
                        res =true;
                pstmt.close();
                fs.close();
        }catch(Exception ex){
                ex.printStackTrace();
        }finally{
                conn.close();
        }
        return res;
}
```

（3）创建 save.jsp 页，在该页中获取图片的路径信息，将图片信息封装到 PictureInfo 对象中，然后调用 PictureDao 类的 saveImage()方法，将图片信息保存到数据库，关键代码如下：

```
<%
request.setCharacterEncoding("UTF-8");
String filePath = request.getParameter("pathStr");              //获取文件路径字符串
PictureInfo picture = new PictureInfo();                        //封装图片信息的 JavaBean 对象
if(filePath!=null&&!filePath.equals("")){
        File file = new File(filePath);                         //根据文件路径创建 File 对象
        String type = file.getName().substring(file.getName().lastIndexOf("."));
        picture.setPictureName(file.getName());                 //图片名称
        picture.setPictureType(type);                           //图片类别
        picture.setPicturePath(file);                           //图片文件
        PictureDao.getInstance().saveImage(picture);            //保存图片
}
%>
```

秘笈心法

由于直接将图片数据保存到数据库会对数据库以及应用程序造成很大的负担，因此笔者不建议采用这种方法保存图片，除非一些重要机密的图片，否则最好不要采取这种方法。可以在数据库中保存图片的路径，而图片只是存储在服务器的磁盘目录中。

实例 268	备份数据库文件 光盘位置：光盘\MR\11\268	高级 实用指数：★★★★

实例说明

数据的备份功能是数据管理系统的重点，它主要涉及两种技术：首先是从数据库中读取需要备份的数据，通常是一个表格；其次是将数据写入到本地文件。本实例通过程序编码也实现了该功能。运行本实例，如图 11.18 所示，在下拉列表中选择需要备份的数据库，然后输入数据库备份文件的路径，单击“备份”按钮后将在指定的路径中生成备份文件。



图 11.18　备份数据库文件

关键技术

实现本实例主要应用到了 Java API 中提供的 Runtime 类和 Process 类。通过 Runtime 对象的 exec()方法执行 MySQLDUMP 命令，并返回一个 Process 实例来控制进程并获得相关信息，使用该实例中提供的 getInputStream()方法获得执行命令所输出的控制台输出流，然后将备份信息写到文件中。

在读取数据库备份数据并写入文件时，主要应用到了 java.io 包中的 InputStreamReader、BufferedReader、FileOutputStream 和 OutputStreamWriter 4 个类。下面对这几个类进行简单介绍。

- ❑ InputStreamReader 类

该类的构造方法如下：

```
public InputStreamReader(InputStream in,String charsetName) throws UnsupportedEncodingException
```

InputStreamReader 使用指定的字符编码读取字节并将其解码为字符。它使用的字符集可以由名称指定或显式给定，或者可以接受平台默认的字符集。在创建 InputStreamReader 对象时，至少应该指定一个 InputStream 类型的参数，InputStream 表示字节输入流的所有类的超类。

- ❑ BufferedReader 类

该类的构造方法如下：

```
public BufferedReader(Reader in)
```

BufferedReader 用于从字符输入流中读取文本。每次调用 InputStreamReader 中的一个 read()方法都会导致从底层输入流读取一个或多个字节。要启用从字节到字符的有效转换，可以提前从底层流读取更多的字节，使其超过满足当前读取操作所需的字节。为了达到最高效率，需要考虑在 BufferedReader 内包装 InputStreamReader。

- ❑ FileOutputStream 类

该类的构造方法如下：

```
public FileOutputStream(File file) throws FileNotFoundException
```

FileOutputStream 俗称文件输出流，它的作用是把内存中的数据输出到文件中。它是一个字节输出流 OutputStream 抽象类下的一个子类。FileOutputStream 用于写入诸如图像数据之类的原始字节的流。

- ❑ OutputStreamWriter 类

该类的构造方法如下：

```
public OutputStreamWriter(OutputStream out)
```

OutputStreamWriter 是字符流通向字节流的桥梁，可使用指定的 charset 将要写入流中的字符编码成字节。它使用的字符集可以由名称指定或显式给定，否则将接受平台默认的字符集。

设计过程

（1）新建用于备份数据库的操作类 BackupDao，在该类中编写获取 MySQL 数据库名称的方法，主要查询

MySQL 数据库系统表 schemata，关键代码如下：

```
/**
 * 查询 schemata 系统数据库表中的所有数据库名
 * @return 返回包含所有数据库名的集合
 */
public List<String> getDatabaseName(){
        List<String> list = new ArrayList<String>();
        Connection conn = null ;
        try{
                conn =DBCon.getConn() ;                                          //创建数据库连接
                String sql = "select schema_name from schemata";                  //查询语句
                PreparedStatement stmt = conn.prepareStatement(sql);              //创建预编译 SQL 对象
                ResultSet rs = stmt.executeQuery();                               //执行查询并返回结果集
                while(rs.next()){
                        list.add(rs.getString(1));                                //所有的数据库名添加到集合中
                }
                rs.close();
        }catch(Exception ex){
                ex.printStackTrace();
        }finally{
                try {
                        conn.close();
                } catch (SQLException e) {
                        e.printStackTrace();
                }
        }
        return list ;
}
```

（2）创建用于备份数据库 backup.jsp 页，在该页中读取所有数据库名，并添加到下拉列表中，关键代码如下：

```
<select name="dbName" id="dataBaseName" class="fontStyle2">
<%
        List<String> list = BackupDao.getInstance().getDatabaseName();
        if(list!=null&&list.size()>0){
                for(String databaseName:list){
%>
        <option value="<%=databaseName %>"><%=databaseName %></option>
        <%}
        }
%>
</select>
```

（3）在 BackupDao 类中编写数据库备份的方法，将数据库信息导出到.sql 文件中，关键代码如下：

```
/**
 * 执行备份的方法
 * @param database 将备份的数据库
 * @param path 备份文件的完整路径名
 * @return 备份成功返回 true，否则返回 false
 */
public boolean backupDB(String database ,String path){
 try{
        String command = "cmd.exe /c mysqldump -uroot -p111 "+database+" ";    //备份命令
        Process p =Runtime.getRuntime().exec(command);                          //执行备份命令
        InputStreamReader isr =new InputStreamReader(p.getInputStream(),"utf8");

        BufferedReader br = new BufferedReader(isr);                            //创建 BufferedReader 对象读取控制台备份信息

        StringBuffer sb = new StringBuffer("");                                 //创建 StringBuffer 对象，用于动态添加每行信息
        String line;
        while((line=br.readLine())!=null){
                sb.append(line+"\r\n");
        }
        FileOutputStream fs = new FileOutputStream(path);                      //创建文件输出流，用于保存备份信息
        OutputStreamWriter writer = new OutputStreamWriter(fs,"utf8");
        writer.write(sb.toString());                                            //将备份数据写入.sql 文件
        writer.flush();
```

```
        isr.close();                                        //关闭相关的文件流
        br.close();
        writer.close();
        fs.close();
        return true;
    }catch(Exception ex){
        ex.printStackTrace();
        return false;
    }
}
```

（4）创建 dobackup.jsp 页，在该页中获取表单请求信息，并调用 BackupDao 类的 backupDB()方法执行数据库备份，关键代码如下：

```
<%
    String databaseName = request.getParameter("dbName");        //数据库名
    String path = request.getParameter("path");                  //备份路径
    BackupDao.getInstance().backupDB(databaseName,path);         //调用备份方法执行备份
%>
```

秘笈心法

本实例中使用的 MYSQLDUMP 命令实现备份的是数据库中所有数据，该命令的语法结构如下：

```
mysqldump –uUser –pPassword DataBase
```

MYSQLDUMP 命令的常用参数及说明如表 11.4 所示。

表 11.4　MYSQLDUMP 命令的常用参数及说明

参　　数	说　　明
-uUser	用户名
-pPassword	密码
-hHost	备份数据库的 IP 地址
-d	只备份表结构而不备份表的数据
-t	只备份表的数据
-F	执行备份命令之前，使用它刷新 MySQL 服务器的 log
--add-locks	在备份文件 Insert 语句中加上 Lock Table 和 UnLock Table 语句，Lock Table 锁定用于当前线程的表，UnLock Table 释放表的锁定
--add-drop-table	在备份文件的 Create 语句之前包含 Drop Table IF EXISTS 语句，避免在导入时出错

实例 269　显示数据库中的图片信息　高级

光盘位置：光盘\MR\11\269　实用指数：★★★★

实例说明

在使用流的方式将图片存入到数据库中之后，该怎样利用存储的数据来还原成原来的图片呢？本实例将展示这个过程，实例运行效果如图 11.19 所示。

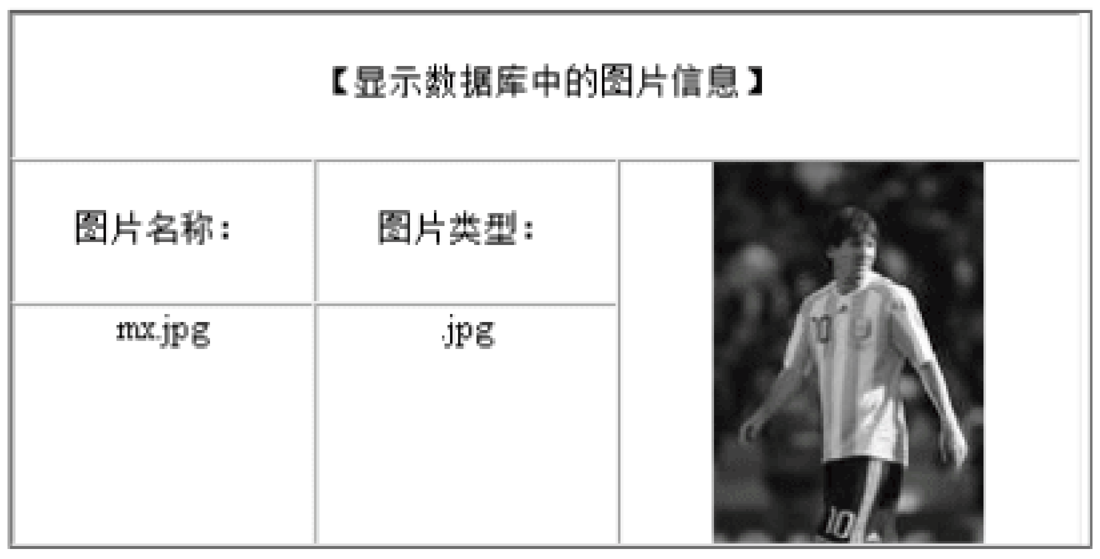


图 11.19　显示数据库中的图片信息

关键技术

在 JSP 页面中将字节信息还原为图片时，需要应用 ServletOutputStream 类的 write()方法，该方法允许将数据以字节的形式输出，ServletOutputStream 对象是通过 response 对象获取的，通过 ServletOutputStream 对象可以响应生成二进制的正文数据。

设计过程

（1）创建封装图片信息的 JavaBean 类 PictureInfo，关键代码如下：

```
public class PictureInfo {
	private int id;                                  //编号
	private String pictureName;                      //图片名称
	private String pictureType;                      //图片类型
	private byte[] picByte;                          //图片的字节
}
```

（2）创建 PictureDao 类，在该类中编写读取图片信息的方法，该方法返回 PictureInfo 类型的对象，关键代码如下：

```
public PictureInfo getImage() throws SQLException{
	Connection conn = null;
	PictureInfo pic = new PictureInfo();             //创建用于封装图片信息的对象
	try{
		conn = DBCon.getConn();                      //创建数据库连接
		String sql = "select * from tb_img where id=2";  //查询 SQL
		Statement stmt = conn.createStatement();     //创建 Statement 对象
		ResultSet rs = stmt.executeQuery(sql);       //执行 SQL 并返回结果集
		if(rs.next()){
			pic.setPictureName(rs.getString("name"));    //图片名称
			pic.setPictureType(rs.getString("type"));    //图片类型
			pic.setPicByte(rs.getBytes("img"));          //图片原始字节
		}
	}catch(Exception ex){
		ex.printStackTrace();
	}finally{
		conn.close();
	}
	return pic;
}
```

（3）创建 img.jsp 页，该页面用于生成图片。首先设置响应正文类型为 image/jpeg，表示生成一个图片，然后调用 PictureDao 类的方法获取图片的字节数据，最后通过 ServletOutputStream 的 write()方法将字节数据输出，关键代码如下：

```
<%@ page contentType="image/jpeg" pageEncoding="utf-8"%>
<%@ page import = "java.io.*" %>
<%@ page import="com.lh.dao.PictureDao" %>
<%@ page import="com.lh.model.PictureInfo" %>
<%
	//设置页面不缓存
	response.setHeader("Pragma","No-cache");
	response.setHeader("Cache-Control","no-cache");
	response.setDateHeader("Expires", 0);
	PictureInfo pic = PictureDao.getInstance().getImage() ;     //调用方法获取图片信息
	OutputStream output=response.getOutputStream();              //获取输出二进制数据的对象
	output.write(pic.getPicByte());                              //输出图片字节数据
	output.flush();
	output.close();
%>
```

（4）创建 index.jsp 页，在该页中添加<img>标签，设置该标签的 src 属性值为 img.jsp，在该页执行时会调用 img.jsp 页生成图片，关键代码如下：

```
<img alt="" src="img.jsp">
```

秘笈心法

显示数据库中图片的过程就是将字节流还原成图片的过程，这个过程涉及很多转换，读者可以将其单独提取出来并制作成一个方法，方便以后使用。

实例 270　读取文件路径到数据库

光盘位置：光盘\MR\11\270

初级

实用指数：★★★★

实例说明

在实例 267 中，实现了将图片存入到数据库的第一种方法，即利用流将文件写入到数据库中。本实例将展示另一种方法，即将图片的路径保存到数据库中，当需要使用图片时，利用路径进行查询。在不必考虑安全性等问题时，使用这种方式更加合理，可以节约大量的磁盘空间，实例运行效果如图 11.20 所示。



图 11.20　读取文件路径到数据库

关键技术

本实例主要根据获取的文件路径来创建 File 对象，然后通过 File 对象的 getAbsolutePath()方法获得文件的绝对路径，最后将文件的路径信息存入到数据库中即可。

设计过程

（1）创建封装图片信息的 JavaBean 类 PictureInfo，关键代码如下：

```
public class PictureInfo {
        private int id;                                          //编号
        private String pictureName;                              //图片名称
        private String pictureType;                              //图片类型
        private String picturePath;                              //图片路径
        …//此处省略了属性的 getXXX()方法和 setXXX()方法
}
```

（2）创建 PictureDao 类，该类主要包含一个将图片信息保存到数据库的方法 saveImage()，关键代码如下：

```
public boolean saveImage(PictureInfo picture) throws SQLException{
        Connection conn = null;
        boolean res = false;
        try{
                conn = DBCon.getConn();                          //获取数据库连接
                String sql = "insert into tb_picture(name,type,path) values(?,?,?)";
                PreparedStatement pstmt = conn.prepareStatement(sql);    //创建预编译对象
                pstmt.setString(1, picture.getPictureName());    //设置图片名称
                pstmt.setString(2, picture.getPictureType());    //设置图片类型
                pstmt.setString(3, picture.getPicturePath());    //设置图片路径
                int i = pstmt.executeUpdate();                   //执行 SQL 语句
                if(i>0)
                        res =true;
                pstmt.close();
        }catch(Exception ex){
                ex.printStackTrace();
        }finally{
                conn.close();
```

```
        }
        return res;
}
```

（3）创建 save.jsp 页，该页用于获取请求，然后调用 PictureDao 类的 saveImage()方法保存图片信息，关键代码如下：

```
<%
        request.setCharacterEncoding("UTF-8");
        String filePath = request.getParameter("pathStr");          //获取文件路径字符串
        PictureInfo picture = new PictureInfo();                    //封装图片信息的 JavaBean 对象
        if(filePath!=null&&!filePath.equals("")){
                File file = new File(filePath);                     //根据文件路径创建 File 对象
                String type = file.getName().substring(file.getName().lastIndexOf("."));
                picture.setPictureName(file.getName());             //设置图片名称
                picture.setPictureType(type);                       //设置图片类别
                picture.setPicturePath(file.getAbsolutePath());     //设置图片路径
                PictureDao.getInstance().saveImage(picture);        //调用保存图片的方法
        }
%>
```

秘笈心法

将图片保存到数据库中有两种方法：第一种是将图片转换成字节流写入到数据库中；第二种是将图片所在的路径写入到数据库，当需要显示图片时，可以利用这个路径来读取。在实际应用中，第二种比较常见，因为可以提高数据库的利用率，也方便读取。

实例 271 在数据库中建立磁盘文件索引

光盘位置：光盘\MR\11\271

初级　实用指数：★★★★

实例说明

用户在使用电脑的过程中，一定使用过查找功能，根据输入的关键字不同，操作系统会在指定的文件夹下进行查找，并返回查找的结果。本实例利用了 File 类将指定的文件夹下（包括子文件夹）所有文件的绝对路径存储到数据库中，再利用 MySQL 的索引功能，方便用户对指定文件进行查询，实例运行效果如图 11.21 所示。



图 11.21　在数据库中建立磁盘文件索引

关键技术

File 类包括了与文件管理系统相关的大量方法，本实例用到的方法如表 11.5 所示。

表 11.5　File 类的常用方法

方　法　名	作　　用
File(String pathname)	通过将给定路径名字符串转换为抽象路径名来创建一个新 File 实例
getAbsoluteFile()	返回此抽象路径名的绝对路径名形式
getAbsolutePath()	以字符串的形式返回该 File 对象的绝对路径
isDirectory()	测试该 File 对象是不是一个文件夹，是则返回 true
listFiles()	如果给定的 File 对象是一个文件夹，则将其转换成 File 数组，数组中包括该文件夹中的文件和子文件夹，否则抛出 NullPointerException

设计过程

（1）创建 FileUtil 工具类，编写获取文件夹中所有文件路径的方法 getFilePath()，参数 list 用于保存迭代出的文件路径，rootFile 用来指明迭代开始的文件夹，关键代码如下：

```
private static List<String> getFilePath(List<String> list, File rootFile) {
    File[] files = rootFile.listFiles();                     //列出用户选择的文件夹下的所有文件（夹）
    if (files == null)                                       //如果是空文件夹直接返回
        return list;
    for (File file : files) {                                //遍历用户选择的文件夹下所有的文件（夹）
        if (file.isDirectory()) {                            //如果是一个文件夹则进行迭代
            getFilePath(list, file);
        } else {
            list.add(file.getAbsolutePath().replace("\\", "/"));    //否则保存路径
        }
    }
    return list;
}
```

技巧：在获取文件的路径过程中，会遇到文件夹中还有文件夹的情况，如果使用迭代就能够简化编码，只需要考虑文件夹和文件两种情况即可。

（2）编写创建磁盘文件索引的方法 createFileIndex()，参数 folderPath 表示要创建索引的磁盘文件夹路径。在该方法中，调用步骤（1）中编写的获取文件夹中所有文件路径的方法 getFilePath()，然后循环该方法返回的路径集合，在循环中定义查询 SQL 语句，调用自定义的查询方法，查询数据库判断是否存在这些文件路径的信息，如果不存在则调用保存信息的方法将这些文件路径保存到数据库，关键代码如下：

```
public static void createFileIndex(File folderPath) throws Exception {
    List<String> list = new ArrayList<String>();             //创建 List 集合
    getFilePath(list, folderPath);              //调用查询所有文件路径的方法，将文件路径保存到 List 集合中
    for (int i = 0; i < list.size(); i++) {                  //循环 List 集合
        String sql = "select id from tb_directories where path = '" + list.get(i) + "';";   //根据文件路径字符串，定义查询 SQL 语句
        int maxId = DBHelper.getMaxID("tb_directories");
        List<Object[]> results = DBHelper.query(sql);        //调用自定义的查询方法
        if (results.size() == 0) {                           //如果没有返回查询结果
            sql = "insert into tb_directories (path) values ('" + list.get(i) + "');";   //定义执行添加信息的 SQL 语句
            DBHelper.update(sql);                            //调用自定义的方法，将文件路径保存到数据库
        }
    }
}
```

（3）编写根据关键字搜索文件的方法 searchFilePathByKey()，参数 key 为用户输入的搜索关键字。在该方法中定义模糊查询的 SQL 语句，调用自定义的数据库查询方法，查询数据库文件索引信息并返回 List 集合，关键代码如下：

```
public static List<Object[]> searchFilePathByKey(String key) {
    String sql = "select * from tb_directories where path like '%" + key + "%';";   //模糊查询的 SQL 语句
    List<Object[]> results = DBHelper.query(sql);            //调用自定义的查询方法，返回 List 集合
    return results;
}
```

说明：数据库操作的相关方法是在 DBHelper 类中定义的，由于这部分内容比较简单，所以这里不进行具体介绍，详细代码请参见本书附带的光盘。

（4）创建 index.jsp 页，在该页中获取用户输入的文件夹路径以及搜索关键字，然后调用 FileUtil 类的方法，在数据库中建立磁盘文件索引，关键代码如下：

```
<%
    List<Object []> filePaths = null;
    request.setCharacterEncoding("UTF-8");                   //设置请求编码
    String filePath = request.getParameter("filepath");      //获取文件夹路径
    String key = request.getParameter("key");                //获取关键字
    if(filePath!=null&&!filePath.equals("")){                //判断是否为空
        FileUtil.createFileIndex(new File(filePath));        //调用创建文件索引的方法
```

```
                filePaths = FileUtil.searchFilePathByKey(key);          //调用文件查询的方法，返回 List 集合
        }
%>
```

（5）在页面的文本域<textarea>中，循环遍历返回查询结果的 List 集合，输出与搜索关键字匹配的文件路径信息，关键代码如下：

```
<textarea rows="5" cols="30" id="fileIndex"><%
    if(filePaths!=null){
        for(Object[] row:filePaths){                           //遍历查询结果的 List 集合
                out.println(row[1]);                           //输出文件路径
        }
    }
%></textarea>
```

秘笈心法

迭代是编程中一种常用的技巧，正确使用迭代能大幅度简化编程。本实例就是使用迭代来获得一个给定文件夹下所有文件（包括子文件夹中的文件）的路径。迭代编程一定要注意退出迭代的条件，否则很可能出现死循环。

实例 272	实现文件简单的加密与解密 光盘位置：光盘\MR\11\272	初级 实用指数：★★★★

实例说明

对磁盘文件进行加密与解密是一项很常见的技术。这样可以保护文件的安全性。通过使用 Java 中的流技术可以很轻松地实现文件的加密与解密。但需要注意的是，对文件实现加密后，必须通过相应的方法才能正确地解密。本实例的运行结果如图 11.22 所示。加密后的文件如图 11.23 所示。

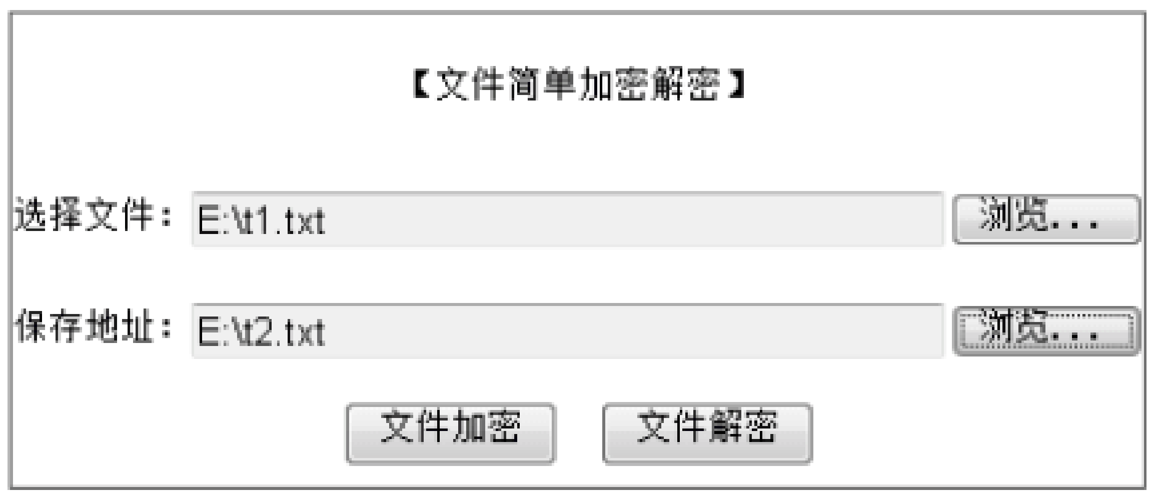


图 11.22　实例的运行结果

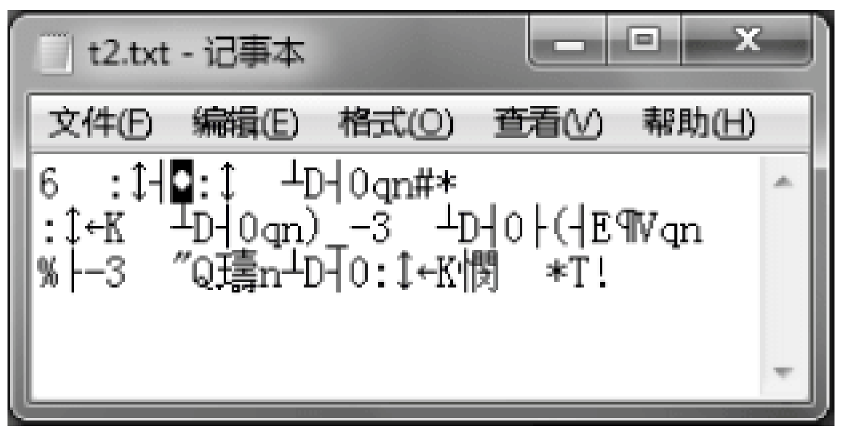


图 11.23　加密后的文件

关键技术

本实例实现的文件加密与解密很简单，就是将通过流从文件中读取的数据进行处理，然后写入到新的文件中，当解密时通过对应的方式对加密的文件进行处理即可。

本实例中对从文件中读取的字节进行处理，从而实现文件加密，关键代码如下：

```
int ibt = buffer[i];
ibt += 100;
ibt %= 256;
```

在对文件实现解密时，再从读取的字节中进行对应的运行，关键代码如下：

```
int ibt = buffer[i];
ibt -= 100;
ibt += 256;
ibt %= 256;
```

设计过程

（1）创建类 EncryptFile，在该类中用于定义文件加密、解密的方法。其中 encry()方法为加密方法，该方法有两个 String 类型的参数，分别用于指定要进行加密的文件路径与加密后文件的保存地址，代码如下：

```
public static boolean encry(String frontFile, String backFile) {
    try {
        File f = new File(frontFile);                                   //根据加密文件地址创建文件对象
        FileInputStream fileInputStream = new FileInputStream(f);       //创建 FileInputStream 对象
        byte[] buffer = new byte[fileInputStream.available()];          //从流中获取可读的字节数
        fileInputStream.read(buffer);                                   //从流中读取字节
        fileInputStream.close();                                        //关闭流
        for (int i = 0; i < buffer.length; i++) {                       //循环遍历从流中读取的数组
            int ibt = buffer[i];
            ibt += 100;                                                 //将数组中的数据做相加运算
            ibt %= 256;
            buffer[i] = (byte) ibt;
        }
        FileOutputStream fileOutputStream = new FileOutputStream(new File(
                backFile));                                             //根据加密后文件的保存地址创建输出流对象
        fileOutputStream.write(buffer, 0, buffer.length);               //向输出流中写数据
        fileOutputStream.close();                                       //将流关闭
        return true;
} catch (Exception e) {
        e.printStackTrace();
         return false;
    }
}
```

（2）创建 unEncry()方法，该方法用于实现文件解密。该方法有两个 String 类型的参数，分别用于指定要进行解密的文件与加密后文件的保存地址，具体代码如下：

```
public static boolean   unEncry(String frontFile, String backFile) {
    try {
        File f = new File(frontFile);                                   //创建要解压的文件对象
        FileInputStream fileInputStream = new FileInputStream(f);       //创建文件输入流对象
        byte[] buffer = new byte[fileInputStream.available()];          //从流中获取可读的字节数
        fileInputStream.read(buffer);                                   //从流中读取字节
        fileInputStream.close();                                        //关闭流
        for (int i = 0; i < buffer.length; i++) {
            int ibt = buffer[i];
            ibt -= 100;                                                 //对从流中读取的数据进行运算处理
            ibt += 256;
            ibt %= 256;
            buffer[i] = (byte) ibt;
        }
        FileOutputStream fileOutputStream = new FileOutputStream(new File(backFile));  //根据要写入的文件地址创建输出流
        fileOutputStream.write(buffer, 0, buffer.length);               //向输出流中写数据
        fileOutputStream.close();                                       //将流关闭
        return true;
      } catch (Exception e) {
        e.printStackTrace();
         return false;
    }
}
```

（3）创建 save.jsp 页，该页用于处理请求，根据请求参数的不同来调用 EncryptFile 类的方法，实现文件的加密和解密，关键代码如下：

```
<%
      request.setCharacterEncoding("UTF-8");
      String sourcePath = request.getParameter("sourcePath");          //获取文件路径字符串
      String targetPath = request.getParameter("targetPath");          //获取文件路径字符串
      String flag = request.getParameter("flag");
      if(sourcePath!=null&&targetPath!=null){
            if(flag!=null&&flag.equals("1")){
                  boolean res = false;
                  res = EncryptFile.encry(sourcePath,targetPath);       //文件加密
            }
            if(flag!=null&&flag.equals("2")){
                  boolean res = false;
                  res = EncryptFile.unEncry(sourcePath,targetPath);     //文件解密
            }
      }
%>
```

秘笈心法

本实例实现文件加密时，主要是对从文件中检索出来的字节进行处理，解密时再通过相应的算法来获取文件的初始字节数据。加密算法是将字节进行加 100 后，对 256 取余。当然，读者也可以根据其他运算实现加密。

实例 273 从 XML 文件中读取数据

光盘位置：光盘\MR\11\273

高级

实用指数：★★★★

实例说明

XML 文件是以节点的形式保存信息，以树形分层结构排列。元素可以嵌套在其他元素中。在 XML 文件中可以保存各种信息，也可以当作数据库来使用。本实例使用 XML 文件保存连接数据库的相关信息，并实现读取 XML 文件中的内容，将其显示在网页中，实例的运行结果如图 11.24 所示。

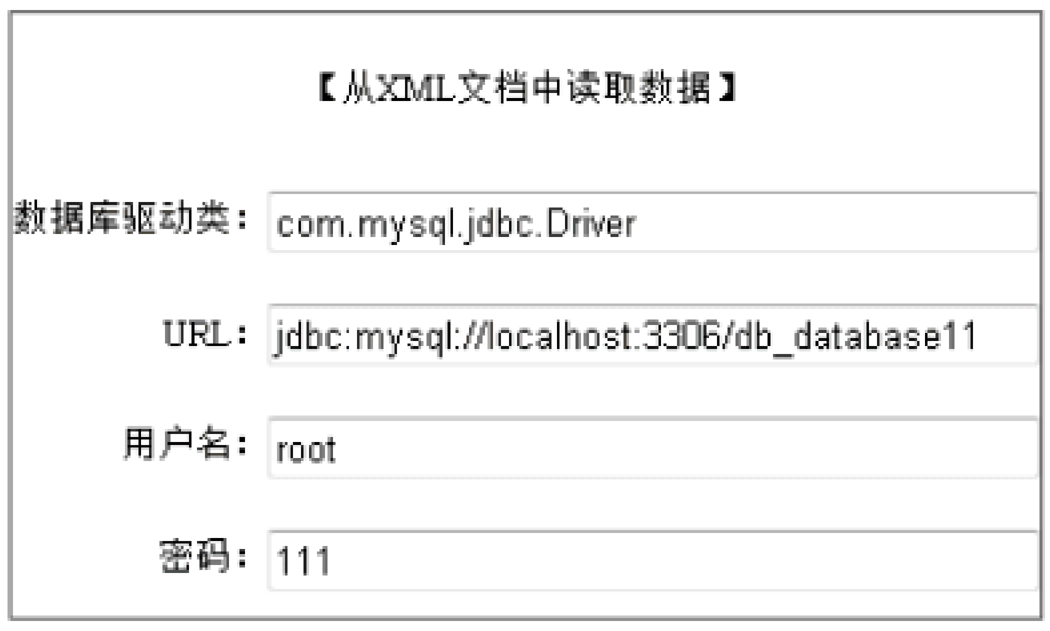


图 11.24　从 XML 文件中读取数据

关键技术

使用开源的 DOM 和 SAX 组件都可以操作 XML 文件。但是都需要下载相关的文件，并将其添加到项目中。为了简便操作，本实例使用 JDK 内置类来实现从 XML 文件中读取数据。实现本实例的功能主要涉及以下几个重要的类与方法。

❑ DocumentBuilderFactory 类

该类表示工厂 API，可以使应用程序能够从 XML 文档获取生成 DOM 对象树的解析器。

❑ DocumentBuilder 类

使用此类可以从 XML 文件读取一个 Document 对象。

❑ Document 接口

该接口表示整个 HTML 或 XML 文档。从概念上讲，该接口表示文档树的根。

通过 Document 接口的 getElementsByTagName()方法，从 XML 文档中读取具有指定标记名称的所有程序元素的有序集合 NodeList 对象，具体语法如下：

```
getElementsByTagName(String tagname)
```

参数说明

tagname：要匹配的标记名称。对于 XML 文件，该参数值是区分大小写的。

设计过程

（1）本实例实现的是显示连接数据库的相关信息，这些信息都是从 XML 文件中读取出来的。创建工具类 ReadXMLData，在该类中定义读取 XML 文件的方法，具体代码如下：

```
public class ReadXMLData {
 private Document document;                         //定义 Document 对象
 private File xmlFile;                              //定义 File 对象
 public ReadXMLData(File xmlFile){                  //带参数的构造方法
        this.xmlFile=xmlFile;
 }
 public   String readXml(String str) {
      DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();//定义从 XML 文档获取生成 DOM 对象的解析器
      try {
            DocumentBuilder builder = factory.newDocumentBuilder();
            document = builder.parse(xmlFile);      //根据 XML 获取 DOM 文档实例
      } catch (Exception e) {
            e.printStackTrace();
```

```
        }
        String subNodeTag = document.getElementsByTagName(str).item(0).getFirstChild().getNodeValue();          //获取指定节点保存的值
        return subNodeTag;                                                                                     //返回读取的信息
    }
}
```

（2）新建 index.jsp 页，在该页中获取 XML 文件的路径，创建 ReadXMLData 类，关键代码如下：

```
<%@ page import="java.io.*" %>
<%@ page import="com.lh.util.ReadXMLData" %>
<%
        String path = application.getRealPath("conn.xml");
        File file = new File(path);
        ReadXMLData readData = new ReadXMLData(file);
%>
```

（3）在 index.jsp 页中调用 ReadXMLData 对象的 readXml()方法，读取 XML 文档数据并赋值给文本框，关键代码如下：

```
<input id="databaseClass" type="text" value="<%=readData.readXml("driverClassName")%>" />
<input id="url" type="text" value="<%=readData.readXml("connection_url")%>" />
<input id="userName" type="text" value="<%=readData.readXml("user")%>" />
<input id="pwd" type="text" value="<%=readData.readXml("password")%>" />
```

秘笈心法

对于数据库连接的相关信息，将其写在 XML 或其他格式的文件中，这样如果项目连接的数据库需要修改，直接修改相应的 XML 文件即可，不用对项目进行修改。因此掌握本实例中介绍的读取 XML 文件的方法是非常有用的。

实例 274	对大文件实现分割处理 光盘位置：光盘\MR\11\274	高级 实用指数：★★★☆

实例说明

大的文件在传输时不太方便，为了便于携带，很多软件都提供了将大的文件进行分割的功能。这样就可以实现将一个较大的文件分割成若干个小的文件，方便携带。运行本实例，如图 11.25 所示，选择一个较大的 zip 文件，并指定分割大小，单击“分割”按钮即可实现文件的分割。



图 11.25　文件分割

说明： 本实例就是将较大的文件分割成若干个小的文件，但是分割后的文件不能作为单独的文件运行。

关键技术

实现本实例的关键是通过输入流读取要分割的文件，再分别从流中读取相应的字节数，将其写入到以 tem 为后缀的文件中。通过 FileInputStream 类的 read()方法可以实现读取文件，该方法的语法格式如下。

（1）语法一：以 byte 数组为参数。表示从输入流中将数组长度或字节读取到 byte 数组中。

```
int read(byte[] b)
```

（2）语法二：从输入流中读取指定的字节到数组中。

```
int read(byte[] b,int off.int len)
```

参数说明

❶ b：存储读取数据的字节数组。

❷ off：目标数组 b 中的开始偏移量。

❸ len：读取的最大字节数。

注意：在使用 read()方法读取字节时，都会抛出 IOException 异常，因此在使用该方法读取字节时要处理该异常。

设计过程

（1）创建工具类 CommunteUtil，在该类中编写实现文件分割的方法，关键代码如下：

```
/**
 * 将大文件进行分割
 * @param commFile 分割文件的地址
 * @param untieFile 分割后文件的保存地址
 * @param filesize 分割文件的大小
 */
public void splitFile(File commFile, File untieFile, int filesize) {
    FileInputStream fis = null;
    int size = 1024 * 1024;                          //用来指定分割文件要以 MB 为单位
    try {
        if (!untieFile.isDirectory()) {              //如果要保存的分割文件地址不是路径
            untieFile.mkdirs();                      //创建该路径
        }
        size = size * filesize;
        int length = (int) commFile.length();        //获取文件大小
        int num = length / size;                     //获取文件大小除以 MB 的得数
        int yu = length % size;                      //获取文件大小与 MB 相除的余数
        String newfengeFile = commFile.getAbsolutePath(); //获取保存文件的完整路径信息
        int fileNew = newfengeFile.lastIndexOf(".");
        String strNew = newfengeFile.substring(fileNew, newfengeFile
                .length());                          //截取字符串
        fis = new FileInputStream(commFile);         //创建 FileInputStream 类对象
        File[] fl = new File[num + 1];               //创建文件数组
        int begin = 0;
        for (int i = 0; i < num; i++) {              //循环遍历数组
            fl[i] = new File(untieFile.getAbsolutePath() + "\\" + (i + 1)
                    + strNew + ".tem");              //指定分割后小文件的文件名
            if (!fl[i].isFile()) {
                fl[i].createNewFile();               //创建该文件
            }
            FileOutputStream fos = new FileOutputStream(fl[i]);
            byte[] bl = new byte[size];
            fis.read(bl);                            //读取分割后的小文件
            fos.write(bl);                           //写文件
            begin = begin + size * 1024 * 1024;
            fos.close();                             //关闭流
        }
        if (yu != 0) {                               //文件大小与指定文件分割大小相除的余数不为 0
            fl[num] = new File(untieFile.getAbsolutePath() + "\\"
                    + (num + 1) + strNew + ".tem");  //指定文件分割后数组中的最后一个文件名
            if (!fl[num].isFile()) {
                fl[num].createNewFile();             //新建文件
            }
            FileOutputStream fyu = new FileOutputStream(fl[num]);
            byte[] byt = new byte[yu];
            fis.read(byt);
            fyu.write(byt);
            fyu.close();
        }
    } catch (Exception e) {
```

```
            e.printStackTrace();
        }
}
```

（2）创建 save.jsp 页，用于获取表单请求信息，并调用 ComminuteUtil 类的文件分割的方法，实现文件分割，关键代码如下：

```
<%
    request.setCharacterEncoding("UTF-8");                              //设置请求编码
    String sourcePath = request.getParameter("sourcePath");              //获取文件路径字符串
    String savePath = request.getParameter("savePath");                  //获取文件的保存路径
    String fileSize = request.getParameter("fileSize");                  //分割文件的大小
    if(sourcePath!=null&&savePath!=null&&fileSize!=null){
        File sourceFile = new File(sourcePath);                          //根据源文件路径创建 File 对象
        File saveFile = new File(savePath);                              //根据保存后的路径创建 File 对象
        int size = Integer.parseInt(fileSize);                           //分割文件的大小
        CominuteUtil.splitFile(sourceFile,saveFile,size);                //调用方法实现文件分割
    }
%>
```

秘笈心法

在程序中获取到的文本框的值都是 String 类型。本实例中 splitFile()方法指定分割文件的大小是 int 类型，可以通过 Integer 对象的 parseInt()方法将字符串类型转换为 int 类型。

实例 275　将分割后的文件重新合并

光盘位置：光盘\MR\11\275

高级　实用指数：★★★☆

实例说明

在实例 274 中为大家介绍了如何实现将较大的文件进行分割，分割后的文件是不能运行的，此时需要通过程序对相应的文件进行重新合并。本实例将介绍如何将分割后的文件进行合并，运行结果如图 11.26 所示。



图 11.26　将分割后的文件进行合并

关键技术

本实例在实现文件合并时，仍然是通过文件字节输入/输出流。在进行文件合并时，需要将要进行合并的所有文件全部读取之后，写入到新文件中。

设计过程

（1）创建写工具类 UniteUtil，在该类中定义文件合并的方法 heBing()，具体代码如下：

```
/**
 * 文件合并
 * @param file      要合并的文件数组
 * @param cunDir  文件保存路径
 * @param hz  合并后文件的格式
 * @return 合并成功返回 true，否则返回 false
 */
public static boolean heBing(File[] file, File cunDir, String hz) {
    try {
        File heBingFile = new File(cunDir.getAbsoluteFile() + "\\UNTIE" + hz); //指定分割后文件的文件名
        if (!heBingFile.isFile()) {
            heBingFile.createNewFile();
        }
        FileOutputStream fos = new FileOutputStream(heBingFile);           //创建 FileOutputStream 对象
        for (int i = 0; i < file.length; i++) {                            //循环遍历要进行合并的文件数组对象
```

```
                FileInputStream fis = new FileInputStream(file[i]);
                int len = (int) file[i].length();                    //获取文件长度
                byte[] bRead = new byte[len];
                fis.read(bRead);                                     //读取文件
                fos.write(bRead);                                    //写入文件
                fis.close();                                         //将流关闭
            }
            fos.close();
            return true;
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
}
```

（2）新建 save.jsp 页，在该页中处理表单请求信息，然后调用 UniteUtil 类的 heBing()方法，实现文件合并，关键代码如下：

```
<%
    request.setCharacterEncoding("UTF-8");                           //设置请求编码
    String sourcePath = request.getParameter("filePath");            //获取文件路径字符串
    String savePath = request.getParameter("savePath");              //获取文件的保存路径
    if(sourcePath!=null&&savePath!=null){
        File sourceFile = new File(sourcePath);                      //根据源文件路径创建 File 对象
        File saveFile = new File(savePath);                          //根据保存后的路径创建 File 对象
        if(!saveFile.exists())
            saveFile.mkdir();
        boolean res = false;
        res = UniteUtil.heBing(sourceFile.listFiles(),saveFile,".zip");
        if(res){%>
            <script type="text/javascript">
                alert("文件合并成功！");
                window.location.href="index.jsp";
            </script>
        <%}
    }
%>
```

秘笈心法

细心的读者可以发现，本实例在 for 循环语句中创建了 FileInputStream 对象，并在 for 循环中将输入流关闭。而 FileOutputStream 对象只创建了一个，是因为要合并成一个文件的小文件有很多个，需要读取每个小文件，就要分别创建 FileInputStream 对象。而合并的文件只有一个，因此只需创建一个 FileOutputStream 对象。

实例 276 利用 StreamTokenizer 统计文件的字符数

光盘位置：光盘\MR\11\276

高级

实用指数：★★★☆

实例说明

在常见的文本编辑器中，有一些提供了对字数的统计，如 Word。但有些文本编辑器是没有提供字数统计的，如记事本工具。为了方便使用记事本的用户可以快速地统计记事本文件中的字符数，可以开发专门的小工具，Java 中的 StreamTokenizer 类可以实现该功能。本实例实现的是统计记事本文件的字符数，运行结果如图 11.27 所示。

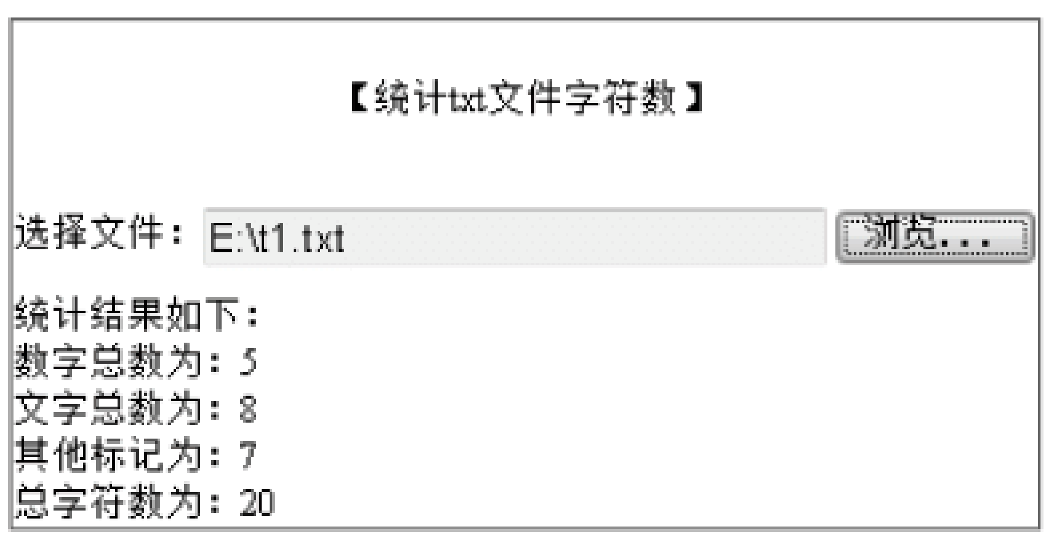


图 11.27　统计文件的字符数

关键技术

java.io 包中的 StreamTokenizer 类可以获取输入流并将其解析为标记，可以通过该类的 nextToken()方法读取下一个标记。该类的构造

方法如下：

```
StreamTokenizer(Reader r)
```

参数说明

r：提供输入流的 Reader 对象。

该类有几个非常重要的常量来标记读取文件的内容，这些常量及说明如表 11.6 所示。

表 11.6　StreamTokenizer 类中的常量及说明

常　量　名	常 量 说 明
TT_EOF	表示读取到文件末尾
TT_WORD	指示读到一个文字标记的常量
TT_NUMBER	表示已读到一个数字标记的常量

设计过程

（1）创建工具类 StatUtil，在该类中定义 statis()方法，获取读取文件的字符数组，该类有一个 String 类型的参数，用于指定文件地址，返回值为保存读取结果的 int 数组，关键代码如下：

```
public static int[] statis(String fileName) {
    FileReader fileReader = null;
    try {
        fileReader = new FileReader(fileName);                          //创建 FileReader 对象
        StreamTokenizer stokenizer = new StreamTokenizer(new BufferedReader(
                fileReader));                                           //创建 StreamTokenizer 对象
        stokenizer.ordinaryChar('\'');                                  //将单引号当作是普通字符
        stokenizer.ordinaryChar('\"');                                  //将双引号当作是普通字符
        stokenizer.ordinaryChar('/');                                   //将“/”当作是普通字符
        int[] length = new int[4];                                      //定义保存计算结果的 int 型数组
        String str;
        int numberSum = 0;                                              //定义保存数字的变量
        int symbolSum = 0;                                              //定义保存英文标点数的变量
        int wordSum = 0;
        int sum = 0;                                                    //定义保存总字符数的变量
        while (stokenizer.nextToken() != StreamTokenizer.TT_EOF) {      //如果没有读到文件的末尾
            switch (stokenizer.ttype) {                                 //判断读取标记的类型
                case StreamTokenizer.TT_NUMBER:                         //如果用户读取的是一个数字标记
                    numberSum += 1;                                     //计算读取的数字长度
                    length[0] = numberSum;                              //设置数组中的元素
                    break;                                              //退出语句
                case StreamTokenizer.TT_WORD:                           //如果读取的是文字标记
                    str = stokenizer.sval;                              //获取该标记
                    wordSum += str.length();                            //计算该文字的长度
                    length[1] = wordSum;
                    break;
                default:                                                //如果读取的是其他标记
                    str = String.valueOf((char) stokenizer.ttype);      //读取该标记
                    symbolSum += str.length();                          //计算该标记的长度
                    length[2] = symbolSum;                              //设置 int 数组中的元素
            }
        }
        sum = symbolSum + numberSum + wordSum;                          //获取总字符数
        length[3] = sum;
        return length;
    } catch (Exception e) {
        e.printStackTrace();
       return null;
    }
}
```

（2）创建 index.jsp 页，在该页中获取表单请求信息中的文件路径，然后调用 StatUtil 类的 statis()方法获取文件的字符数，关键代码如下：

```
<%@ page import="java.io.*"  %>
<%@ page import="com.lh.util.StatUtil" %>
<%
	request.setCharacterEncoding("UTF-8");                  //设置请求编码
	String filePath = request.getParameter("sourcePath");   //获取文件路径
	int count[]=new int[4];                                 //创建 int 数组
	if(filePath!=null&&!filePath.equals("")){               //判断字符串是否为空
		count = StatUtil.statis(filePath);                  //调用方法获取文件的字符数
	}
 %>
```

（3）在 index.jsp 页的表格中输出数组变量 count 中的元素值，关键代码如下：

```
<td align="left" colspan="2">
	统计结果如下：<br/>
	数字总数为：<%= count[0]%><br/>
	文字总数为：<%=count[1] %><br/>
	其他字符为：<%=count[2] %><br/>
	总字符数为：<%=count[3] %><br/>
</td>
```

秘笈心法

在统计文件的字符数时，不能简单地统计标记数，因为字符数不等于标记，按照标记的规定，引号中的内容 10 页算是一个标记。所以要使引号的内容都算作一个标记，就要使用 StreamTokenizer 类的 ordinaryChar()方法将单引号和双引号当作普通字符处理。

实例 277	序列化与反序列化对象 光盘位置：光盘\MR\11\277	高级 实用指数：★★★☆

实例说明

对于一个大的应用程序需要使用很多的对象，由于虚拟机内存有限，有时不可能将所有有用的对象都放在内存中，因此，需要将不常用的对象暂时持久化到文件中，这一过程就称为对象的序列化；当需要使用对象时，再从文件中把对象恢复到内存，这个过程被称为对象的反序列化。本实例实现的是将对象序列化到文件，然后再从文件反序列化到对象，实例运行结果如图 11.28 所示。



图 11.28　实例运行结果

关键技术

需要被序列化的对象必须实现 java.io.Serializable 接口，需要注意的是该接口中没有定义任何的方法。对象输出流 ObjectOutputStream 可以将对象写入到流中，该类的构造方法如下。

- ❑ 语法一：创建写入指定 OutputStream 的 ObjectOutputStream。

```
Object OutputStream(OutputStream out)
```

参数说明

out：要写入数据的输出流。

- ❑ 语法二：完全实现 ObjectOutputStream 的子类提供的方法。

```
Object OutputStream()
```

对象输入流 ObjectInputStream 类可以从流中读取对象到内存，该类的构造方法如下。

- ❑ 语法一：创建从指定 InputStream 读取的 ObjectInputStream。

```
Object InputStream(InputStream in)
```

参数说明

in：要从中读取的输入流。

❑ 语法二：完全实现 ObjectInputStream 的子类。

```
ObjectInputStream()
```

设计过程

（1）创建工具类 SerializeObject，在该类中定义内部类 Bowel，表示被序列化的对象，该类实现了 Serializable 接口，具体代码如下：

```
static class Bowel implements Serializable{
    private int number1,number2;                                     //定义普通的实例变量
    private transient int number3;                                   //定义不会被序列化和反序列化的对象
    private static int number4;
    public Bowel(int number1 ,int number2,int c,int number3){        //构造方法
        this.number1 = number1;
        this.number2 = number2;
        this.number3 = number3;
        this.number4 = number4;
    }
}
```

（2）在该类中定义 serialize()方法，该方法用于实现对象的序列化，在该方法中通过 ObjectOutputStream 类的 writeObject()方法将对象写入到文件中，具体代码如下：

```
public static void serialize(String fileName){
    try {
        File file = new File(fileName);                              //根据文件地址创建文件对象
        if(!file.exists()){                                          //如果该对象不存在
            file.createNewFile();                                    //创建该文件对象
        }
        ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(fileName));  //创建对象输出流对象
        out.writeObject("今天是:");                                  //向文件中写入对象数据
        out.writeObject(new Date());
        Bowel my1 = new Bowel(5,6,7,3);                              //定义内部类对象
        out.writeObject(my1);                                        //将对象写入到文件中
        out.close();                                                 //将流关闭
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

（3）创建 deserialize()方法用于实现对象的反序列化，具体代码如下：

```
public static Object[] deserialize(String fileName){
    try {
        File file = new File(fileName);                              //根据文件地址创建文件对象
        if(!file.exists()){                                          //如果该文件不存在
            file.createNewFile();                                    //新建文件
        }
        ObjectInputStream in = new ObjectInputStream(new FileInputStream(fileName));  //创建对象输入流
        String today = (String)(in.readObject());                   //从流中读取对象信息
        Date date = (Date)(in.readObject());                        //从流中读取对象信息
        Object[] object = {today,date};                             //将读取出的对象添加到对象数组中
        Bowel my1 = (Bowel)(in.readObject());
        in.close();                                                  //关闭流
        return object;
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}
```

秘笈心法

在对象序列化时，对象是按照 writeObject()方法的调用顺序存储在文件中的，先被序列化的对象在文件的前面，后被序列化的对象在文件的后面。因此，在反序列时，先读到的对象就是先被序列化的对象。

11.2　无组件的文件上传

文件上传几乎是所有网站都具有的功能，用户可以将文件上传到服务器的指定文件夹中，也可以保存在数据库中。下面将介绍几个无组件的文件上传的实例。

实例 278　单表单元素上传文件到数据库

光盘位置：光盘\MR\11\278

高级　实用指数：★★★☆

实例说明

在 Web 应用中，用户经常需要上传个人资料文件或图片。本实例将介绍如何将指定文件上传到服务器的数据库中。运行本实例，如图 11.29 所示，单击“浏览”按钮选择要上传的文件，单击“提交”按钮，文件将被保存到数据库中。



图 11.29　单表单元素上传文件到数据库

关键技术

本实例使用自定义的 UploadBean 类来解析 Form 表单的 Input Stream 输入流，并从中获取文件信息。在 UploadBean 类中使用类输入流对象的 read()方法来读取文件的字节数据。

设计过程

（1）创建工具类 UploadBean，在该类中主要包含一个用于解析 Form 表单的方法，表单类型为 multipart/form-data，并且将数据存放在 Map 集合类的实例对象中，关键代码如下：

```
public class UploadBean {
private InputStream InputStreamState;                          //输入流
private HttpServletRequest request;                            //从 JSP 页面传入的 Request
private Map<String, String> parameter = new HashMap<String, String>(); //创建 Map 集合
/**
 * 解析 Form 表单
 */
private void resolverForm() {
        InputStream inputStream = null;
        try {
                inputStream = request.getInputStream();        //获取请求的输入流
        } catch (IOException e2) {
                e2.printStackTrace();
        }
        String contentType = request.getContentType();         //获取请求正文类型
        if (contentType != null && inputStream != null) {
                InputStreamState = inputStream;
                int data;
                StringBuffer datastr = new StringBuffer();     //创建 StringBuffer 对象
                parameter.clear();                             //移除 Map 集合中的映射关系
                try {
                        while ((data = inputStream.read()) != -1) {   //从输入流中读取单个字节
                                datastr.append((char) data);   //将数据添加到 StringBuffer 对象中
                        }
                        inputStream.close();                   //关闭输入流
                        String split = "boundary=";
                        String splitStr = "--"
                                        + contentType.substring(contentType.indexOf(split)
                                                        + split.length());
                        String[] formFileds = datastr.toString().split(
                                        "Content-Disposition: form-data; ")   //分隔表单数据
                        for (int i = 0; i < formFileds.length; i++) {   //解析表单数据
```

```
                        int[] index = new int[4];
                        if (!formFileds[i].startsWith(splitStr)) {
                            index[0] = -1;
                            index[1] = formFileds[i].indexOf("\n", index[0]);
                            index[2] = formFileds[i].indexOf("\n", index[1] + 1);
                            index[3] = formFileds[i].indexOf("\n", index[2] + 1);
                            String name = "";
                            for (int lc = 0; lc < index.length - 1; lc++) {
                                String line = formFileds[i].substring(
                                        index[lc] + 1, index[lc + 1]);
                                String[] lineFields = line.split("; ");
                                for (int j = 0; j < lineFields.length; j++) {
                                    if (lineFields[j].startsWith("name=")) {
                                        name = lineFields[j].substring(
                                                lineFields[j].indexOf("\"") + 1,
                                                lineFields[j].lastIndexOf("\""));
                                    }
                                    if (j > 0) {
                                        String arg = name
                                                + "_"
                                                + lineFields[j].substring(0,
                                                        lineFields[j].indexOf("="));
                                        String argContent = lineFields[j]
                                                .substring(lineFields[j]
                                                        .indexOf("\"") + 1,
                                                        lineFields[j]
                                                                .lastIndexOf("\""));
                                        parameter.put(arg, argContent);
                                    }
                                }
                                if (line.equals("\r")) {
                                    parameter.put(name, formFileds[i].substring(
                                            index[lc + 1] + 1, formFileds[i]
                                                    .lastIndexOf(splitStr) - 2));
                                    break;
                                }
                            }
                        }
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
}
…//此处省略了属性的 getXXX()方法和 setXXX()方法
}
```

（2）创建数据库操作类 FileUploadDao，在该类中编写将信息保存到数据库的方法，关键代码如下：

```
public boolean   saveFile(String file) throws SQLException{
        Connection conn = null;
        boolean res = false;
        try{
            conn = DBCon.getConn();                              //创建数据库连接
            String sql = "insert into tb_file(file) values(?)";        //SQL 语句
            PreparedStatement stmt = conn.prepareStatement(sql); //创建预编译对象
            stmt.setBytes(1, file.getBytes("ISO-8859-1"));          //设置文件的原始字节
            int i = stmt.executeUpdate();                         //执行 SQL 命令
            if(i>0)
                res = true;
        }catch(Exception ex){
            ex.printStackTrace();
        }finally{
            conn.close();
        }
        return res;
}
```

（3）创建 save.jsp 页，在该页中调用 UploadBean 类的方法解析 Form 表单数据，然后调用 FileUploadDao 类的 saveFile()方法保存文件到数据库，关键代码如下：

```
<%@ page import="com.lh.dao.FileUploadDao" %>
<%@ page import="com.lh.model.UploadBean" %>
<%
	UploadBean uploadBean = new UploadBean();                    //创建 UploadBean 对象，用于解析表单数据
	uploadBean.setRequest(request);                              //将请求传入 UploadBean 对象
	FileUploadDao.getInstance().saveFile(myFile);                //调用方法保存到数据库
%>
```

秘笈心法

由于在 UploadBean 类中需要获取请求的输入流 InputStream，所以在调用 UploadBean 类的方法解析 Form 表单数据时，需要将 JSP 页面的 request 对象传入到 UploadBean 中，然后在 UploadBean 中再根据请求对象进行相应的处理。这样可以避免将大量的 Java 代码编写在 JSP 页面中，有利于程序的维护。

实例 279 多表单元素上传文件到数据库

光盘位置：光盘\MR\11\279

高级

实用指数：★★★☆

实例说明

在网站中的用户注册页面中，除了基本的用户信息以外，可能还会包含上传用户照片的表单元素，用户注册后会将用户信息以及用户的照片保存到数据库中。本实例实现的就是这一功能，运行本实例，如图 11.30 所示，输入用户信息并选择用户照片，单击“提交”按钮后，用户信息以及照片将保存到数据库中。

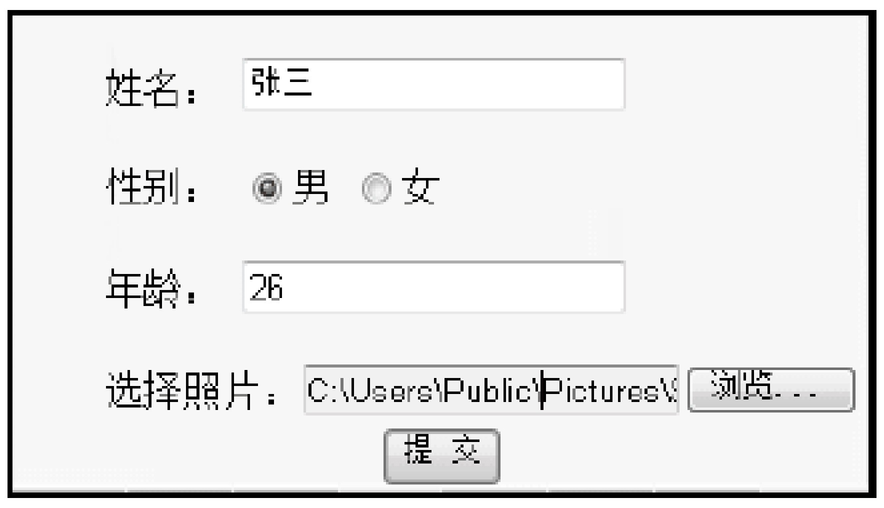


图 11.30　多表单元素上传文件到数据库

关键技术

本实例同样是设置 Form 表单的提交类型为 multipart/form-data，然后根据请求的输入流来解析表单元素的数据，包括文件数据。从 InputStream 输入流中解析表单数据的方法与实例 278 中 UploadBean 类的解析方法相同，此处不再赘述。

设计过程

（1）创建 JavaBean 类 UserInfo，用于封装用户的注册信息，关键代码如下：

```
public class UserInfo {
	private String userName;                          //姓名
	private String userSex;                           //性别
	private int userAge;                              //年龄
	private String userPic;                           //照片
…//省略了属性的 getXXX()方法和 setXXX()方法
}
```

（2）创建数据库操作类 FileUploadDao，在该类中编写保存用户注册信息的方法，关键代码如下：

```
public boolean    saveFile(UserInfo user) throws SQLException{
	Connection conn = null;
	boolean res = false;
	try{
		conn = DBCon.getConn();                                                  //创建数据库连接
		String sql = "insert into tb_user(name,sex,age,pic) values(?,?,?,?)";    //SQL 语句
		PreparedStatement stmt = conn.prepareStatement(sql);                     //创建 PreparedStatement 对象
		stmt.setString(1, user.getUserName());
		stmt.setString(2, user.getUserSex());
		stmt.setInt(3, user.getUserAge());
		stmt.setBytes(4, user.getUserPic().getBytes("ISO-8859-1"));              //设置文件的原始字节
		int i = stmt.executeUpdate();                                            //执行 SQL 命令
		if(i>0)
			res = true;
	}catch(Exception ex){
```

```
            ex.printStackTrace();
        }finally{
            conn.close();
        }
        return res;
}
```

（3）创建 save.jsp 页，在该页中调用 FileUploadUtil 类的方法解析 Form 表单数据，然后调用 FileUploadDao 类的 saveFile()方法保存用户注册信息到数据库，关键代码如下：

```
<%@ page import="com.lh.dao.FileUploadDao" %>
<%@ page import="com.lh.model.UserInfo" %>
<%@ page import="com.lh.util.FileUploadUtil" %>
<%
 FileUploadUtil uploadUtil = new FileUploadUtil();
 uploadUtil.setRequest(request);                          //将请求对象传入到 FileUploadUtil 对象中
 String name = uploadUtil.getParameter("name");
String nameStr = new String(name.getBytes("ISO-8859-1"),"UTF-8");
 String sex = uploadUtil.getParameter("sex");
 String age = uploadUtil.getParameter("age");
 String myFile = uploadUtil.getParameter("file");
 UserInfo user = new UserInfo();                          //封装用户信息的 JavaBean 对象
 user.setUserName(nameStr);                               //设置姓名
 user.setUserSex(sex);                                    //设置性别
 user.setUserAge(Integer.parseInt(age));                  //设置年龄
 user.setUserPic(myFile);                                 //设置照片
FileUploadDao.getInstance().saveFile(user);               //调用方法保存到数据库
%>
```

秘笈心法

在实际的项目开发中，为了便于维护以及提示代码的重用性，一般会将表单数据封装到一个单独的 JavaBean 对象中。例如，本实例中将表单的信息的用户名、年龄、性别以及照片封装到 UserInfo 对象中，而这个 UserInfo 对象中的属性和数据库中用户信息表的字段是对应的，这种方式类似于 ORM 映射（Object Relation Mapping，对象关系映射）。目前，体现 ORM 映射最好的框架之一就是 Hibernate，Hibernate 框架主要用于持久化对象，有兴趣的读者可以参考相关 Hibernate 的资料，此处不作详细介绍。

实例 280	上传文件到服务器 光盘位置：光盘\MR\11\280	高级 实用指数：★★★☆

实例说明

大多数网站都有文件上传的功能，如 CSDN 网站允许用户将一些程序的源代码进行上传，以便于其他用户下载。本实例将介绍如何将文件上传到服务器的指定目录中。运行本实例，如图 11.31 所示，选择要上传的文件，单击“上传”按钮后，所选文件将会上传至服务器。



图 11.31　上传文件到服务器

关键技术

本实例在实现上传时，同样需要解析类型为 multipart/form-data 的 Form 表单数据，通过这种类型提交的表单数据，需要从请求的 InputStream 输入流中进行解析。

接下来就是利用 java.io.FileOutputStream 将解析出的文件数据写入到服务器的文件中。FileOutputStream 俗称文件输出流，它的作用是把内存中的数据输出到文件中，该类中包含一个 write()方法，其语法结构如下：

```
public void write(byte[] b) throws IOException
```

参数说明

b：向文件输出流中输出的字节数组。

功能：将 b.length 个字节从指定 byte 数组写入此文件输出流中。

设计过程

（1）创建工具类 FileUploadUtil，编写上传文件到服务器的方法，具体代码如下：

```
/**
 * 上传文件到服务器
 * @param file 要上传的文件
 * @param filename 文件名
 * @param uploadPath 上传路径，此路径是服务器的路径
 * @return 上传成功返回 true，否则返回 false
 */
public boolean uploadToServer(String file, String filename, String uploadPath) {
	filename = System.currentTimeMillis()+ "_"+filename;		//文件重命名
	resolverForm();
	try {
		File dir = new File(uploadPath+"/upload/");
		if(!dir.exists())					//测试此目录是否存在
			dir.mkdir();				//创建文件夹
		File newFile = new File(dir.getAbsolutePath()+"/"+filename);	//创建文件
		if(!newFile.exists())				//测试此文件是否存在
			newFile.createNewFile();		//创建文件
		FileOutputStream fos = new FileOutputStream(newFile);	//创建文件输出流对象
		fos.write(file.getBytes("iso-8859-1"));		//向文件输出流中输出文件字节数据
		fos.close();					//关闭流
		return true;
	} catch (Exception ex) {
		ex.printStackTrace();
		return false;
	}
}
```

（2）创建 save.jsp 页，在该页中获取请求数据，然后将请求传入 FileUploadUtil 对象中，通过 FileUploadUtil 对象的方法来解析 Form 表单的数据，最后调用 FileUploadUtil 对象的上传文件的方法，实现将文件上传到服务器，关键代码如下：

```
<%@ page import="com.lh.util.FileUploadUtil" %>
<%@ page import="java.io.*" %>
<%@page import="java.net.URLDecoder"%>
<%
	FileUploadUtil uploadUtil = new FileUploadUtil();
	uploadUtil.setRequest(request);				//将请求对象传入到 FileUploadUtil 对象中
	String filePath = uploadUtil.getParameter("pathStr");	//文件路径字符串
	filePath=URLDecoder.decode(filePath,"UTF-8");		//对编码过的字符串进行解码
	File file = new File(filePath);				//根据文件路径创建 File 对象
	String fileName = file.getName();			//获取文件名
	String uploaFile = uploadUtil.getParameter("file");	//要上传的文件
	String uploadPath = application.getRealPath("/");	//服务器路径作为上传路径
	uploadUtil.uploadToServer(uploaFile,fileName,uploadPath);
%>
```

秘笈心法

大多数网站的文件上传功能，一般都是将文件上传到服务器的指定目录，而不是将文件保存到数据库中，除非一些重要的保密文件。如果将所有文件都保存到数据库中，不仅会给数据库造成非常大的负担，而且也会影响程序的性能。因此，笔者建议不要将上传文件保存到数据库，将其上传至服务器即可。

实例 281	限制文件大小的文件上传 光盘位置：光盘\MR\11\281	高级 实用指数：★★★☆

实例说明

本实例实现的是对实例 280 的改进，限制了上传文件的大小。运行本实例，如图 11.32 所示，限制文件大

小不能超过 2MB，如果上传文件超过 2MB，Tomcat 服务器会弹出文件超出大小限制的异常提示信息。



图 11.32　限制文件大小的文件上传

关键技术

本实例的基本实现思路和前几个实例基本相同，唯一的不同就是在上传文件时需要判断文件的大小，如果文件超出指定大小，设置应用程序抛出一个自定义的异常提示信息。在程序中抛出自定义的异常，语法格式如下：

```
throw new Exception(String str)
```

参数说明

str：自定义异常的错误提示信息。

功能：抛出一个自定义的异常对象。

设计过程

（1）创建 FileUploadUtil 类，在该类中添加一个 long 类型的属性 fileSize，用该属性表示文件的大小，并对该属性添加 getXXX()方法和 setXXX()方法。然后在上传文件的方法 uploadToServer()中，添加文件大小验证的代码，限制上传文件的大小。uploadToServer()方法的关键代码如下：

```
public boolean uploadToServer(String file, String filename, String uploadPath)
 throws Exception {
	if (file.length() > fileSize) {						//判断文件大小是否超过指定的值
		file = null;
		errorMessage = "文件超出大小限制。";
		parameter.put("error", errorMessage);
		throw new Exception("文件超出大小限制。");
	}
	filename = System.currentTimeMillis()+ "_"+filename;			//文件重命名
	resolverForm();
	try {
		File dir = new File(uploadPath+"/upload/");
		if(!dir.exists())							//测试此目录是否存在
			dir.mkdir();						//创建文件夹
		File newFile = new File(dir.getAbsolutePath()+"/"+filename);	//创建文件
		if(!newFile.exists())						//测试此文件是否存在
			newFile.createNewFile();				//创建文件
		FileOutputStream fos = new FileOutputStream(newFile);		//创建文件输出流对象
		fos.write(file.getBytes("iso-8859-1"));				//向文件输出流中输出文件字节数据
		fos.close();							//关闭流
		return true;
	} catch (Exception ex) {
		ex.printStackTrace();
		return false;
	}
}
```

（2）在 save.jsp 页中添加设置文件大小的代码，主要是调用 FileUploadUtil 的 setXXX()方法对 fileSize 属性赋值，注意代码中的粗体部分的两行代码为设置文件大小的代码。save.jsp 页的关键代码如下：

```
<%
	FileUploadUtil uploadUtil = new FileUploadUtil();
	uploadUtil.setRequest(request);
	long fileSize = 1024*1024*2;						//设置文件大小为 2MB
	uploadUtil.setFileSize(fileSize);
```

```
        String filePath = uploadUtil.getParameter("pathStr");            //文件路径字符串
        filePath=URLDecoder.decode(filePath,"UTF-8");                    //对编码过的字符串进行解码
        File file = new File(filePath);                                  //根据文件路径创建 File 对象
        String fileName = file.getName();                                //获取文件名
        String uploaFile = uploadUtil.getParameter("file");              //文件
        String uploadPath = application.getRealPath("/");                //服务器路径作为上传路径
        uploadUtil.uploadToServer(uploaFile,fileName,uploadPath);
%>
```

秘笈心法

上传文件的功能是需要限制文件大小的，具体限制值为多少需要根据服务器性能而定。如果用户上传一个几 GB 的文件甚至更大的文件，那么服务器端在读取并写入这些数据时，如果处理不当会严重影响服务器的运行效率，而且这个文件也会占用服务器系统很大的磁盘空间。

11.3　通过组件实现文件上传

在 11.2 节中讲解了无组件上传的几个实例，实现无组件上传时，需要编写大量的代码来实现具体的上传功能。本节将向大家介绍如何应用第三方的开源组件实现文件上传，目前比较常用的上传组件是 jspSmartUpload 组件和 commons-fileupload 组件，应用它们可以不必编写大量的代码，只是简单地调用即可，具体的业务实现是在组件中完成的。

实例 282　使用 jspSmartUpload 组件实现文件上传　高级

光盘位置：光盘\MR\11\282　　实用指数：★★★☆

实例说明

jspSmardUpload 上传组件是比较常用的上传组件。下面将向读者介绍使用 jspSmardUpload 上传组件实现文件的上传，实例运行效果如图 11.33 所示。



图 11.33　使用 jspSmardUpload 组件实现文件上传

关键技术

在实现本实例前，先介绍一下 jspSmardUpload 组件，这个组件中有 3 个核心类，下面介绍这几个类的用法。

1. File 类

该类包含以下几个主要方法。

- getCount()方法：获取上传文件的数目，返回值是 int 型。
- getSize()方法：获取上传文件的总长度，单位为字节，返回值为 long 型。
- getFile()方法：有一个 int 型参数，用于获取参数指定位置处的 com.jspsmart.uploadFile 对象。
- getCollection()方法：将所有 File 对象以 Collection 的形式返回。
- getEnumeration()方法：将所有 File 对象以 Enumeration 的形式返回。

2. Request 类

因为实现文件上传时，表单属性 enctype 设置成了 multipart/form-data，这样只有通过组件提供的 Request 类才可以获取表单元素，该类包含以下几个主要方法。

- getParameter()方法：获取表单中由参数指定的表单元素的值。
- getParameterName()方法：获取 Form 表单中除<input type=“file”>外的所有表单元素的名称，其返回

值的类型是枚举类型的。

- getParameterValues()方法：获取的是表单中具有相同名称的元素的值。方法中传入的参数的类型是一个 String 类型，参数是用来指定表单中元素的名称。

3. SmartUpload 类

该类主要用于实现文件的上传下载，在实现时要首先实现 initialize()方法，然后可以分别使用下面的常用方法。

- upload()方法：实现本方法是一个复杂的过程，首先要调用 JSP 内置对象的 getInputStream()方法获取客户端输入流，然后使用输入流的 read()方法读取所有数据写到字节数组中，之后循环提取每个文件的数据并将其封装到 File 对象中，最后通过 Files 类的 addFile()方法添加到 Files 对象中。
- save()方法：在实现了 initialize()方法和 upload()方法后，通过本方法将全部的上传文件保存到指定的目录下面，并返回保存的文件个数。

设计过程

（1）首先实现一个上传文件的页面 upl.html，在实现这个页面时一定要注意的是，要将表单元素中 enctype 的值改为 multipart/fom-data，具体的实现代码如下：

```
<body>
    <form method="post" action="/Jsup/upfile.jsp" enctype="multipart/form-data">
            <p>上传的文件是：(文件大小不要超过 10KB)<br>
            <input type="file" name="file1" size="50"><br>
            <input type="submit" value="Upload">
            </p>
      </form>
</body>
```

（2）创建 upfile.jsp 页，在该页中获取文件上传数据，然后保存到服务器指定目录下。在该页中只需调用组件提供的方法即可，并不需要再去创建自己的实现方法，但一定要注意包的引用，具体代码如下：

```
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
<%@ page import="com.jspsmart.upload.*" %>
<jsp:useBean id="mySmartUpload" scope="page" class="com.jspsmart.upload.SmartUpload" />
<%
    mySmartUpload.initialize(pageContext);              //执行初始化操作
    mySmartUpload.setTotalMaxFileSize(100000);          //设置文件最大字节数
    mySmartUpload.upload();                             //上传文件到服务器
    try{
        mySmartUpload.save("/upload");                  //如果存在目录就保存文件，否则抛出异常
        out.print("成功上传文件！  ");
    }
    catch(Exception e){
        out.print(e.toString());
    }
%>
```

秘笈心法

如果对上传功能的要求不大，读者完全可以直接去使用组件提供的一些方法，不需要大量地编写代码，这样可以在很大程度上提高效率。

实例 283	使用 jspSmartUpload 组件实现中文名文件上传	高级
	光盘位置：光盘\MR\11\283	实用指数：★★★☆

实例说明

jspSmartUpload 上传组件是比较常用的上传组件。但是在使用时如果有对中文文件名的文件进行上传，那么就要对上面的实现方法进行相应的改进。本实例将介绍如何应用 jspSmartUpload 组件处理中文名文件的上传，

实例运行效果如图 11.34 所示。



图 11.34　使用 jspSmartUpload 组件实现中文名文件上传

关键技术

jspSmartUpload 组件没有对中文进行处理，如果希望该组件支持中文，需要对该组件的源代码进行改进。但是由于其官网 www.jspsmart.com 目前已经停用，所以暂时没有 jspSmartUpload 组件的源码文件，不过可以利用 Java 反编译器将 jar 包进行反编译，然后就可以修改其源代码。在组件的源码中，主要修改 SmartUpload 类的 upload()方法和 getDataHeader()方法。

提示：Java 反编译器可以将.class 文件重新编译为.java 源代码文件。目前有近 30 多种 Java 反编译器，如 Java Decompiler、JAD、JODE 等。这些工具可以到其官网下载使用。

设计过程

（1）在 SmartUpload 类中找到 upload()方法，在最后一个 else 语句中，在将字节数组转换为 String 字符串时，设置以 UTF-8 格式进行转换。当然，如果 request 请求为其他编码格式，如 GB2312，那么在 upload()方法中同样设置编码为 GB2312。upload()方法修改之后的关键代码如下：

```
public void upload() throws ServletException, IOException, SmartUploadException{
    ……//此处省略了其他源码
    for (this.m_currentIndex += 1; this.m_currentIndex < this.m_totalBytes; this.m_currentIndex += 2)
    {
      String s1 = getDataHeader();
      this.m_currentIndex += 2;
      boolean flag3 = s1.indexOf("filename") > 0;
      String s3 = getDataFieldValue(s1, "name");
      if (flag3)
      {
         ……//此处省略了其他源码
      }
      getDataSection();
      if ((flag3) && (s4.length() > 0))
      {
        ……//此处省略了其他源码
      }
      if (flag3)
      {
       ……//此处省略了其他源码
      }
      else {
        //源代码中没有设置 UTF-8 编码，所以需要在此处设置 UTF-8 编码
        String s11 = new String(this.m_binArray, this.m_startData, this.m_endData - this.m_startData + 1,"UTF-8");
        this.m_formRequest.putParameter(s3, s11);
      }
      if ((char)this.m_binArray[(this.m_currentIndex + 1)] == '-')
        return;
    }
  }
```

（2）在 SmartUpload 类中找到 getDataHeader()方法，在该方法中同样设置返回字符串的编码格式，关键代

码如下：

```
private String getDataHeader(){
    ……//省略了以上其他源代码
      String header="";
      try {
            header = new String(this.m_binArray, i, j - i + 1,"UTF-8");//在此处设置返回字符串的编码格式为 UTF-8
      } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
      }
      return header;
}
```

（3）在操作系统的 cmd.exe 命令行工具中运行 jar 命令将修改过的 jspSmartUpload 组件源代码打包成 jar 文件，然后将 jar 包复制到项目的 WEB-INF/lib 目录中即可，在命令行中为文件打包的命令如下：

```
jar cvf jspsmartupload_zh_CN.jar com
```

秘笈心法

如果不想修改 jspSmartUpload 组件源代码来解决中文名文件的上传问题，那么就必须将文件上传的页面修改为 html 文件，应用 html 文件上传时的中文文件名同样不会出现乱码。

实例 284　应用 jspSmartUpload 组件处理文件上传漏洞

光盘位置：光盘\MR\11\284

高级　实用指数：★★★☆

实例说明

网站提供的上传文件功能，允许客户端用户将一些文件复制到服务器中，并且用户可访问上传的文件，那么其安全性就出现在这里，如果用户上传了一些 EXE 可执行文件或者 JSP 文件，这些文件可能是用户专门编写的具有获取服务器重要信息甚至对服务器进行破坏功能的恶意文件，这些文件被执行后会产生严重后果。所以在实现文件上传的功能时，可限制用户上传一些具有危险性的文件，如*.exe、*.jsp、*.bat 等。本实例将通过 jspSmartUpload 组件处理文件上传的漏洞，实例运行的效果如图 11.35 所示，选择一个.exe 文件，单击“上传”按钮，此时文件上传失败，并在页面中显示受限制上传文件类型的提示信息。



图 11.35　文件上传页面（左），显示限制文件类型的提示信息（右）

关键技术

实现文件类型上传的限制，主要是应用 jspSmartUpload 组件中的 SmartUpload 类，在该类中提供了一个用于设置禁止上传的文件方法 setDeniedFilesList()，该方法的语法格式如下：

```
public void setDeniedFilesList(String deniedFilesList) throws ServletException, IOException, SQLException
```

参数说明

deniedFilesList：指定禁止上传文件的扩展名，多个扩展名之间以逗号分隔。

功能：设置禁止上传的文件类型，调用该方法时会抛出 ServletException、IOException 和 SQLException。若禁止上传没有扩展名的文件，以“,,”表示。例如，setDeniedFilesList("exe,jsp,,bat")表示禁止上传*.exe、*.jsp、*.bat 和不带扩展名的文件。

设计过程

（1）创建选择文件上传页面，实现代码如下：

```
<form action="doup.jsp" method="POST" enctype="multipart/form-data">
<table border="1" height="200" width="450" bordercolor="gray" bordercolorlight="gray"
bordercolordark="white" cellspacing="0" rules="none">
 <tr bgcolor="#B3DC38" height="35"><td align="center" colspan="3"><%=errors%></td></tr>
        <% for(int i=1;i<4;i++){ %>
        <tr>
            <td align="right" width="20%">文件<%=i%>：</td>
            <td align="center"><input type="file" name="file<%=i%>" size="35"></td>
        </tr>
        <% } %>
        <tr bgcolor="#B3DC38" height="30">
         <td align="center" colspan="2">
                <input type="submit" value="上传">
                <input type="reset" value="重置">
            </td>
        </tr>
</table>
</form>
```

（2）表单提交给 doup.jsp 进行处理，在 doup.jsp 页面中将应用 jspSmartUpload 组件上传文件，实现代码如下：

```
<%@ page contentType="text/html;charset=gb2312"%>
<%@ page import="com.jspsmart.upload.File" %>
<%@ page import="com.jspsmart.upload.Files" %>
<jsp:useBean id="myup" class="com.jspsmart.upload.SmartUpload"/>
<center>正在上传文件，请稍等……</center>
<%
String filedir="/file/";                                    //上传目标文件夹
String errors="";
long maxsize=2*1024*1024;                                   //上传文件的最大空间限制
boolean allow=true;
try{
        myup.initialize(pageContext);                       //初始化
        myup.setMaxFileSize(maxsize);                       //设置文件的最大空间限制
        myup.setDeniedFilesList("exe,jsp,bat");             //设置限制的文件
        myup.upload();                                      //执行上传方法
}catch(SecurityException e){
        allow=false;
        errors+="禁止上传\"*.exe\"、\"*.jsp\"、\".bat\"文件";  //上传失败提示信息
        e.printStackTrace();
}
if(allow){                                                  //如果上传成功
     try{
            Files files=myup.getFiles();                    //获取上传的文件对象
            for(int i=0;i<files.getCount();i++){            //遍历文件
                    File singlefile=files.getFile(i);
                    if(!singlefile.isMissing()){
                            String name=singlefile.getFileName();      //获取上传文件的文件名
                            singlefile.saveAs(filedir+name,File.SAVEAS_VIRTUAL);
                            errors+="<li>文件"+(i+1)+"上传成功！</li>";  //上传成功的信息
                    }
            }
     }catch(java.lang.NumberFormatException e){
            errors="文件上传失败！";
            e.printStackTrace();
     }
  }
request.setAttribute("errors",errors);                      //保存提示信息
%>
<jsp:forward page="fileup.jsp"/>
```

秘笈心法

应用 jspSmartUpload 组件实现文件上传时，除了调用 SmartUpload 类的 setDeniedFilesList()方法禁止上传

的文件外，还可以通过 setAllowedFilesList(String allowedFilesList)方法来设置只允许上传的文件，其中参数 allowedFilesList 指定允许上传文件的扩展名，多个扩展名之间以逗号分隔。如果需要允许上传没有扩展名的文件，以“,,”表示。例如，setAllowedFilesList("txt,doc,,")表示只允许上传*.txt、*.doc 和不带扩展名的文件。

实例 285　使用 commons-fileUpload 组件实现文件上传　高级

光盘位置：光盘\MR\11\285　实用指数：★★★☆

实例说明

commons-fileUpload 上传组件是 apache 的一个开源项目，该组件对中文进行了良好的处理，也就是说，应用该组件上传文件不会出现中文乱码问题，是目前应用最广泛的开源组件。该组件包文件可以到 apache 的官方网站进行下载。本实例就是通过 commons-fileUpload 组件实现文件的上传，实例运行效果如图 11.36 所示。

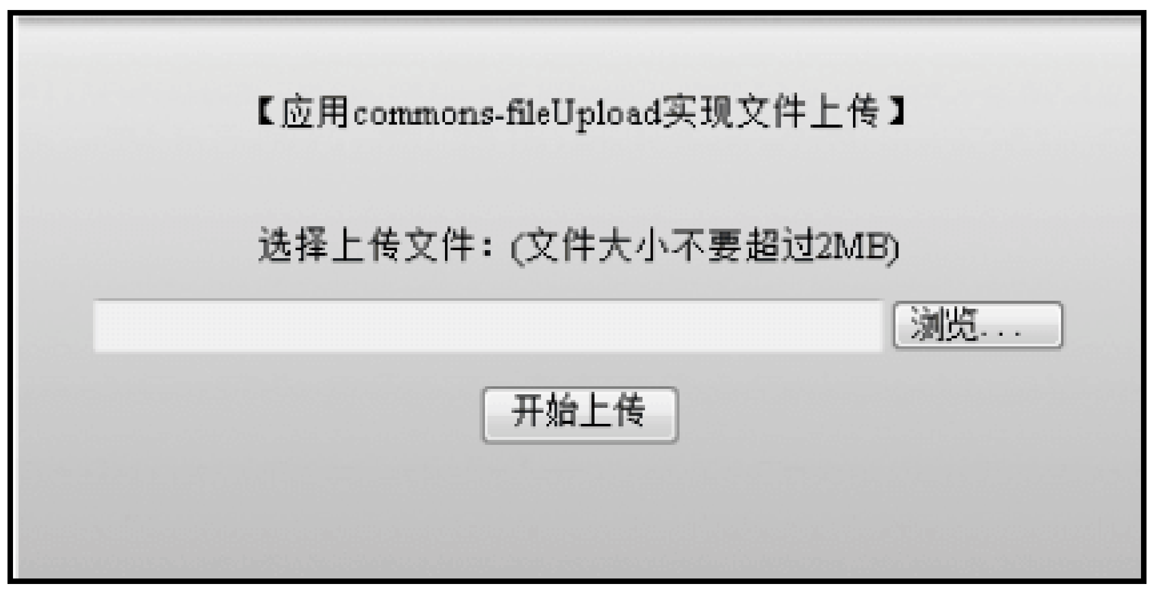


图 11.36　使用 commons-fileUpload 组件实现文件上传

关键技术

commons-fileUpload 组件需要 commons-io 包的支持，在使用该组件时要加以注意。首先需要应用该组件中 ServletFileUpload 类的 isMultipartContent()方法判断请求是否为上传文件的请求，只有是文件时才执行下面的一些相关操作。ServletFileUpload 类是 commons-fileUpload 组件处理文件上传的核心类，该类的常用方法如表 11.7 所示。

表 11.7　ServletFileUpload 类的常用方法

方　法　名	作　　用
public boolean isMultipartContent(HttpServletRequest request)	返回 boolean 值 true 或 false，用于判断请求是否为上传文件的请求，主要是判断 form 表单提交请求类型是否为 multipart/form-data
public List parseRequest(HttpServletRequest request)	该方法从请求中获取上传文件域的 List 集合
public Long getFileSizeMax()	获取 FileItem 对象文件大小的最大值，返回值为 Long 类型。FileItem 对象为 parseRequest()方法获取的 List 集合中的元素
public void setFileSizeMax(Long fileSizeMax)	设置 FileItem 对象文件大小的最大值，设置的参数为 Long 类型。FileItem 对象为 parseRequest()方法获取的 List 集合中的元素
public FileItemIteratorgetItemIterator(HttpServlet Request request)	该方法从请求中获取文件的迭代器

设计过程

（1）创建 index.html 页，在该页中添加一个文件域的表单，并设置提交类型为 multipart/form-data，具体代码如下：

```
<body>
    <form   method="post" action="UploadServlet" enctype="multipart/form-data">
        <p>上传的文件是：(文件大小不要超过 2MB)<br>
        <input type="file" name="file1" size="50"><br>
```

```
            <input type="submit" value="Upload">
        </form>
</body>
```

（2）创建获取文件上传请求的 Servlet 类。在 Servlet 类的 doPost()方法中获取请求，实现文件上传，具体代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)    throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        String uploadPath=getServletContext().getRealPath("/")+"upload";       //定义上传文件的地址
        File folder = new File(uploadPath);
        if(!folder.exists())
                folder.mkdirs();
        try{
                if(ServletFileUpload.isMultipartContent(request)){              //判断获取的是否是文件
                        DiskFileItemFactory disk=new DiskFileItemFactory();
                        disk.setSizeThreshold(20*1024);                         //设置内存可存字节数
                        disk.setRepository(disk.getRepository());               //设置临时文件目录
                        ServletFileUpload up=new ServletFileUpload(disk);
                        int maxsize=2*1024*1024;
                        List list=up.parseRequest(request);                     //获取上传列表
                        Iterator i=list.iterator();                             //创建列表的迭代器
                        while(i.hasNext()){
                                FileItem fm=(FileItem)i.next();                 //遍历列表
                                if(!fm.isFormField()){
                                        String filePath = fileItem.getName();   //获取文件全路径名
                                        String fileName="";
                                        int startIndex = filePath.lastIndexOf("\\");
                                        if(startIndex!=-1){                     //对文件名进行截取
                                                fileName = filePath.substring(startIndex+1);
                                        }else{
                                                fileName=filePath;
                                        }
                                        if(fm.getSize()>maxsize){
                                                message="文件太大了，不要超过 2MB";
                                                break;
                                        }
                                        if((fileName==null)||( fileName.equals(""))&&(fm.getSize()==0)){
                                                message="文件名不能为空，文件大小也不能为零！";
                                                break;
                                        }
                                        File saveFile=new File(uploadPath, fileName);
                                        fm.write(saveFile);                     //向文件中写入数据
                                        message="文件上传成功！";
                                }
                        }
                }
        }catch(Exception ex){
                ex.printStackTrace();
        }
        request.setAttribute("result",message);
        request.getRequestDispatcher("message.jsp").forward(request, response);
}
```

秘笈心法

使用 Commons-fileUpload 组件时，在表单页面每一个元素的属性中都要写好 name 这个属性，否则 commons-fileUpload 组件是不做处理的。

实例 286	通过 commons-fileUpload 组件获取其他表单元素 光盘位置：光盘\MR\11\286	高级 实用指数：★★★☆

实例说明

本实例实现了获取表单的元素值的操作。由于将表单的属性 enctype 设置成 multipart/form-data，这样通过

request 对象的 getParameter()方法就无法获取，所以要做特别的处理。本实例将介绍如何应用 commons-fileUpload 组件获取其他表单元素，实例运行效果如图 11.37 所示。

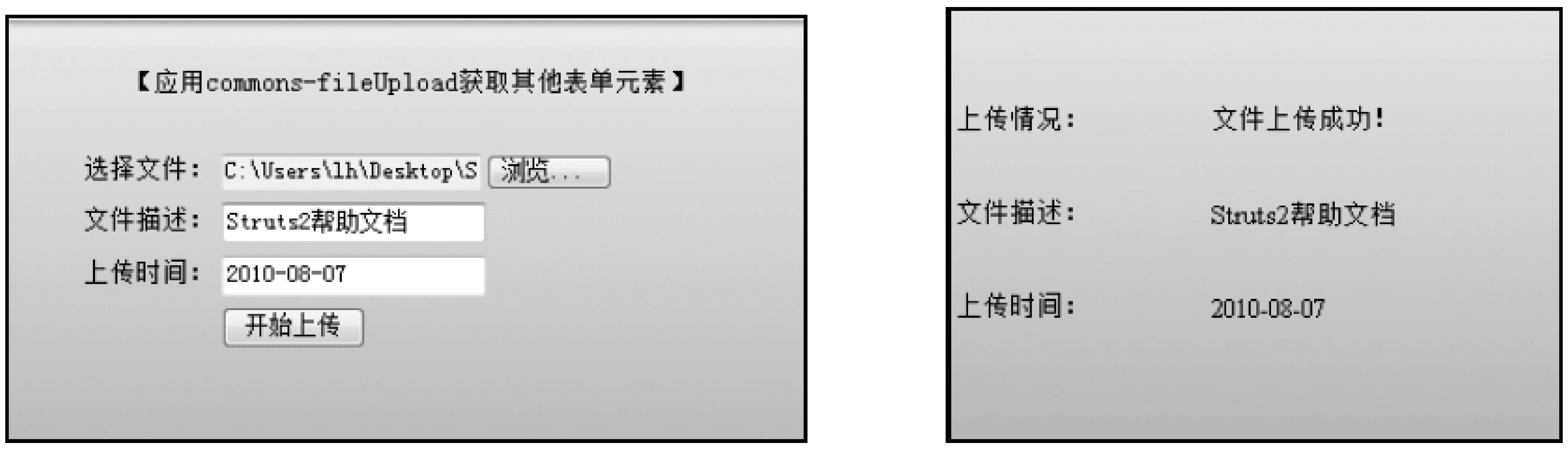


图 11.37　使用 commons-fileUpload 组件实现获取表单元素

关键技术

commons-fileUpload 组件在进行表单处理时，需要使用 ServletFileUpload 类的 parseRequest(request)方法获取上传文件的 List 集合，然后使用 isFormFile()方法判断是普通的表单属性还是一个文件，如果是普通的表单属性，则可以通过以下方法获取表单元素：

```
fileItem.getFieldName();
```

功能：获取表单元素的名称，返回值为 String 类型。

获取表单元素的值可以应用 FileItem 对象的 getString()方法。在通过 getString()方法获取表单元素的值时，为了避免出现乱码，可以设置值的编码格式，代码如下：

```
String user="";
if(fileItem.getFieldName().equals("userName"))
        user = fileItem.getString("UTF-8");
```

设计过程

（1）创建 index.jsp 页，在页面中添加一个文件上传的表单，并且在表单中添加关于文件描述的属性信息，具体代码如下：

```
<body>
<%!
    Date now=new Date();
    String form=String.format("%tF",now);
%>
      <form   method="post" action="upload" enctype="multipart/form-data">
                <p>上传的文件是：(文件大小不要超过 2MB)<br>
                <input type="file" name="file1" size="50"><br>
                <p>文件描述
                <input type="text" name="upDe" size="50"><br>
                <p>上传时间
                <input type="text" name="uptime" size="50" value=<%=form%>><br>
                <input type="submit" value="提交">
        </form>
</body>
```

（2）创建处理上传文件请求的 Servlet 实现类。在 Servlet 类的 doPost()方法中，从请求中获取上传文件的 List 集合，然后迭代集合中的元素，判断元素是普通表单项还是文件，如果是文件则保存到服务器指定目录下，如果是普通表单元素，则获取表单元素的值并保存到 request 对象域中。doPost()方法的具体代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String uploadPath=this.getServletContext().getRealPath("/");          //定义上传文件的地址
        String message=null,content=null,dtme=null;
        try{
                if(ServletFileUpload.isMultipartContent(request)){            //判断获取的是否是文件
                        DiskFileItemFactory disk=new DiskFileItemFactory();
                        disk.setSizeThreshold(20*1024);                       //设置内存可存字节数
                        disk.setRepository(disk.getRepository());             //设置临时文件目录
                        ServletFileUpload up=new ServletFileUpload(disk);
                        int maxsize=2*1024*1024;
```

```
                List list=up.parseRequest(request);             //获取上传列表
                Iterator i=list.iterator();                     //创建列表的迭代器
                while(i.hasNext()){
                    FileItem fm=(FileItem)i.next();             //遍历列表
                    if(!fm.isFormField()){
                        String filePath = fileItem.getName();   //获取文件全路径名
                        String fileName="";
                        int startIndex = filePath.lastIndexOf("\\");
                        if(startIndex!=-1){                     //对文件名进行截取
                            fileName = filePath.substring(startIndex+1);
                        }else{
                            fileName=filePath;
                        }
                        if(fm.getSize()>maxsize){
                            message="文件太大了，不要超过 2MB";
                            break;
                        }
                        String fileSize=new Long(fm.getSize()).toString();
                        if((fileName ==null)||( fileName.equals(""))&&(fileSize.equals("0"))){
                            message="文件名不能为空，文件大小也不能为零！";
                            break;
                        }
                        File saveFile=new File(uploadPath, fileName);
                        fm.write(saveFile);                     //向文件中写入数据
                        message="文件上传成功！";
                    }
                    else{
                        String foename=fm.getFieldName();       //获取表单元素名
                        String con=fm.getString("gbk");         //获取表单内容，注意编码方式
                        //表单元素
                        if(foename.equals("upDe")){
                             content = con;
                        }
                        else if(foename.equals("uptime")){
                             dtme = con;
                        }
                    }
                }
            }
        }
        catch(Exception ex){
            ex.printStackTrace();
        }
        request.setAttribute("result",message);
        request.setAttribute("upDe",content);
        request.setAttribute("dtme",dtme);
        request.getRequestDispatcher("message.jsp").forward(request, response);
}
```

（3）创建 message.jsp 页，在该页中获取并显示所有表单中的元素内容，具体代码如下：

```
<body>
        <table border=1>
            <tr><td>上传情况：</td><td>
        <%=request.getAttribute("result")%></td></tr>
        <tr><td>上传情况：</td><td>
            <%=request.getAttribute("upDe")%></td></tr>
        <tr><td>上传时间：</td><td>
        <%=request.getAttribute("dtme")%></td></tr>
       </table>
</body>
```

秘笈心法

对于 FileItem 对象的 getString()方法有两种重载方式：第一种是无参的，编码方式是系统默认的；第二种是有参的，参数就是定义的编码方式，如 UTF-8、GBK 等。为了避免获取请求数据时出现中文乱码，可以应用设置编码格式的 getString()方法。

实例 287	通过 commons-fileUpload 组件限制上传文件类型 光盘位置：光盘\MR\11\287	高级 实用指数：★★★☆

实例说明

如果上传文件的类型是可执行文件，可能对服务器造成安全隐患，因此需要对上传文件的类型进行限制。本实例将介绍如何应用 commons-fileUpload 组件限制上传文件的类型，实例运行效果如图 11.38 所示，当上传文件类型为*.exe、*.jsp 或*.bat 时，将弹出禁止上传的提示信息。





图 11.38　通过 commons-fileUpload 组件限制上传文件类型

关键技术

本实例主要应用的是 commons-io 组件中 org.apache.commons.io.filefilter 包中的 SuffixFileFilter 类，该类主要用于过滤文件的后缀名，也就是过滤文件类型。SuffixFileFilter 类实现自 java.io.FileFilter 接口，该类的方法及说明如表 11.8 所示。

表 11.8　SuffixFileFilter 类的常用方法及说明

方　法	说　明
public SuffixFileFilter(String suffix)	构造方法，参数 suffix 表示要过滤的文件后缀字符串
public SuffixFileFilter(String[] suffixes)	构造方法，参数 suffixes 表示要过滤的文件后缀字符串数组
public SuffixFileFilter(List suffixes)	构造方法，参数 suffixes 表示要过滤的文件后缀字符串集合
public boolean accept(File file)	过滤方法，参数 file 表示要进行过滤的文件对象。如果 file 对象的文件类型与构造方法中指定的字符串 suffix、字符串数组 suffixes 或集合 suffixes 表示的后缀相同，返回 true，否则返回 false

设计过程

创建处理上传文件请求的 Servlet 实现类。在 Servlet 类的 doPost()方法中，从请求中获取上传文件的 List 集合，然后应用文件后缀过滤器类 SuffixFileFilter 对上传文件的类型进行过滤，如果上传文件的类型为*.exe、*.bat 或*.jsp，则禁止上传。doPost()方法的具体代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
	String uploadPath = this.getServletContext().getRealPath("/")+"upload";	//定义上传文件的服务器路径
	File uploadFolder = new File(uploadPath);	//根据该路径创建 File 对象
	if(!uploadFolder.exists())	//如果路径不存在，则创建
		uploadFolder.mkdirs();
	String message = "文件上传成功！";
	try{
		if(ServletFileUpload.isMultipartContent(request)){
			DiskFileItemFactory factory = new DiskFileItemFactory();	//创建磁盘工厂，用来配置上传组件 ServletFileUpload
			factory.setSizeThreshold(20*1024);	//设置内存中允许存储的字节数
			factory.setRepository(factory.getRepository());	//设置存放临时文件的目录
			ServletFileUpload upload = new ServletFileUpload(factory);	//创建新的上传文件句柄
			int maxSize = 5*1024*1024;	//定义上传文件的大小
```

```
                List<FileItem> files = upload.parseRequest(request);        //从请求中得到所有上传域列表
                String[] suffixs =new String[]{".exe",".bat",".jsp"};         //定义限制上传的文件类型的字符串数组
                SuffixFileFilter filter = new SuffixFileFilter(suffixs);     //创建文件后缀过滤器
                for(FileItem fileItem:files){                                //遍历上传文件集合
                    if(!fileItem.isFormField())  {                           //忽略其他不是文件域的所有表单信息
                        String name = fileItem.getName();
                        String filePath = fm.getName();                      //获取文件全路径名
                        String fileName="";
                        int startIndex = filePath.lastIndexOf("\\");
                        if(startIndex!=-1){                                  //对文件名进行截取
                            fileName = filePath.substring(startIndex+1);
                        }else{
                            fileName=filePath;
                        }
                        if(fileItem.getSize()>maxSize){                      //限制文件大小
                            message = "上传文件不得超过 5MB！";
                            break;
                        }
                        if((fileName == null) ||( fileName.equals(""))&&(fileItem.getSize()==0))
                            continue;
                        File file = new File(uploadPath, fileName);          //在上传路径中创建文件对象
                        boolean res = filter.accept(file);                   //调用文件后缀过滤器的过滤方法，对上传文件类型进行过滤
                        if(res){
                            message = "禁止上传 *.exe、*.jsp、*.bat 文件！";
                            break;
                        }else{
                            fileItem.write(file);                            //向文件写数据
                        }
                    }
                }
            }
        }
        catch(Exception ex){
            ex.printStackTrace();
        }
        request.setAttribute("result", message);                             //将提示信息保存在 request 对象中
        request.getRequestDispatcher("index.jsp").forward(request, response);
}
```

秘笈心法

在 org.apache.commons.io.filefilter 包中，还包含很多文件过滤器类，如 SizeFileFilter（文件大小过滤器）、PrefixFileFilter（文件名前缀过滤器）、CanReadFileFilter（只读文件过滤器）等，这些过滤器都实现了 java.io.FileFilter 接口。如果需要对文件的其他属性进行过滤，可以应用这些过滤器来处理。

11.4 文 件 下 载

在互联网中基本可以查询到所有想要的资料，有些资料信息会非常庞大，如游戏、电影、音乐、软件等，它们从几十兆的字节到几百兆的字节不等，这就需要将文件下载到本地并保存。下面通过几个实例介绍实现文件下载的功能。

实例 288	利用响应输出流实现文件下载 光盘位置：光盘\MR\11\288	高级 实用指数：★★★☆

实例说明

目前在很多网站中都提供影视信息和常用软件的下载，为用户提供帮助的同时达到吸引用户、增加访问量的目的。本实例将介绍如何应用响应流来实现文件下载。运行本实例，单击要下载文件的下载图标，即可弹出

“文件下载”对话框，如图 11.39 所示，单击“保存”按钮即可将文件保存到指定位置。



图 11.39　利用响应输出流实现文件下载

关键技术

本实例主要通过设置响应头信息和输出流来实现。在 JSP 中设置响应头信息时，使用的是 JSP 内置对象 response 对象的 addHeader()方法，该方法的语法格式如下：

```
response.addHeader(String head,String value)
```

参数说明

❶ head：指定响应头的属性名称，如内容类型、内容长度等。

❷ value：指定属性名称对应的属性值。

设计过程

（1）创建 index.jsp 页，在页面中添加一个文件下载的超链接。此时需要通过 ServletContext 对象的 getRealPath()方法获得文件在服务器上的真实路径，关键代码如下：

```
<td>
        <div align="center">
        <A HREF="download.jsp?path=<%=getServletContext().getRealPath("DSC00143.jpg") %>">
                <img src="images/download.GIF" width="22" height="22" border="0">
        </A>
       </div>
</td>
```

（2）创建 download.jsp 页，在该页中实现文件下载功能。根据 index.jsp 页利用超链接提交过来的请求路径信息，创建 File 文件对象，然后将响应头设置为文件下载格式，再通过 ServletOutputStream 输出流将文件以流的方式输出，关键代码如下：

```
<%@ page language="java" import="java.util.*,java.io.*"       pageEncoding="GBK"%>
<%
        response.setCharacterEncoding("utf-8");                 //设置响应编码格式
        String path=request.getParameter("path");               //获取请求参数的值，该参数为文件的服务器路径
        path=new String(path.getBytes("iso-8859-1"));
        File file = new File(path);                             //根据文件路径创建 File 对象
        InputStream in = new FileInputStream(file);             //创建文件输入流，读取文件内容
        OutputStream os = response.getOutputStream();           //获取响应输出流
        //设置响应头
        response.addHeader("Content-Disposition", "attachment;filename="+ new String(file.getName().getBytes("gbk"),"iso-8859-1"));
        response.addHeader("Content-Length", file.length() + "");
        response.setContentType("application/octet-stream");    //设置响应正文类型
        int data = 0;
        while ((data = in.read()) != -1) {                      //从文件流中循环读取字节
                os.write(data);                                 //输出字节流
        }
```

```
        os.close();
        in.close();
%>
```

秘笈心法

在获得参数 path（即文件的路径信息）时，将编码转换成 GBK 编码以支持中文的文件名和路径。在设置响应头中的文件名称时，需要将文件名再从 GBK 转换为 ISO-8859-1，这样在浏览器的下载窗口中才会提示中文的文件名。

实例 289 防止网站文件盗链下载

光盘位置：光盘\MR\11\289

高级

实用指数：★★★☆

实例说明

网站文件盗链下载是指未经允许而私自将其他网站中的一些资源的链接地址（如文章、图片、音乐、软件等地址）嵌入在自己的网站中，从而不必将这些资源上传到自己的空间中仍可以提供给用户浏览和下载。盗链可能导致被盗链网站的服务器因访问的人数过多而瘫痪。本实例将介绍一种防止网站文件被盗链下载的方法，运行本实例，如图 11.40 所示，如果网站被盗链，当单击盗链的超链接下载文件时将弹出错误提示信息。

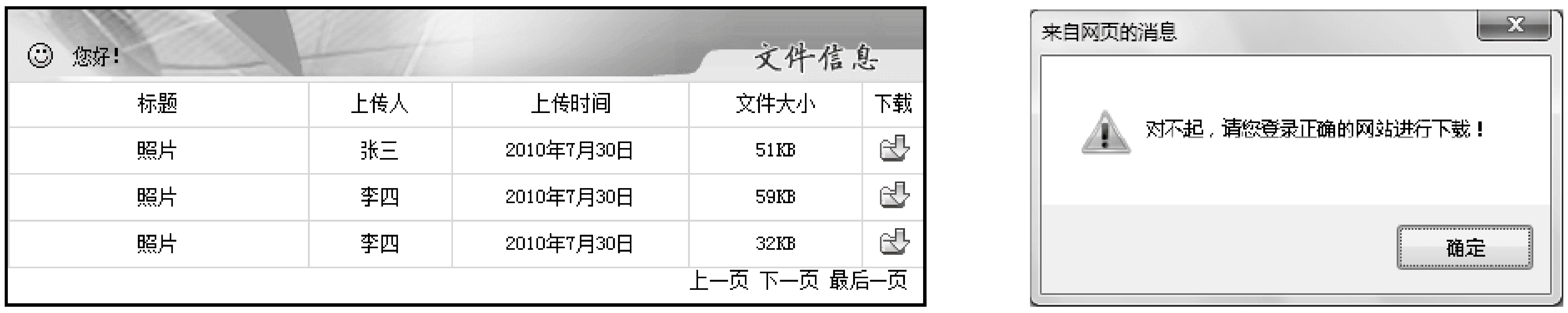


图 11.40　防止网站文件盗链下载

关键技术

防止网站文件盗链下载主要应用的是 request 对象中的 getHeader()方法，获取请求中 referer 字段的值，该字段中记录了当前请求的上一次访问的地址。本实例就是通过这个值来判断当前的请求是否是从其他网站中触发的，从而做出相应的处理。getHeader()方法的语法如下：

```
public String getHeader(String name)
```

功能：返回 HTTP 请求头部的特定项。如果没有指定的请求头参数，则返回 null。

参数说明

name：指定请求头的参数名。

设计过程

（1）创建 index.jsp 页，为存储在服务器指定文件夹下的文件设置下载超链接，链接到 download.jsp 文件，在该文件中完成下载。其中主要应用 getServletContext()方法和 getRealPath()方法，关键代码如下：

```
<td><div align="center">
        <A HREF="download.jsp?path=<%=getServletContext().getRealPath("测试图片.jpg") %>">
        <img src="images/download.GIF" width="22" height="22" border="0"></A>
</div></td>
```

（2）创建 download.jsp 文件完成下载，并应用 request 请求中的 getHeader()方法获取请求中字段 referer 的值，通过判断 referer 字段中记录的当前请求的上一次访问的地址与指定的地址是否相同，从而判断出是否是盗链，然后再进行相应的处理，关键代码如下：

```
<%@ page language="java" import="java.util.*,java.io.*"     pageEncoding="GBK"%>
<%
        String from = request.getHeader("referer");  //获取当前请求的上一次访问的地址
```

```
    //判断当前请求的上一次访问的地址与指定的地址是否相同
    if ((from == null) || (from.indexOf("localhost:8080/262/") < 0)) {
        out.print("<script>alert('对不起，请您登录正确的网站进行下载！');window.location.href='http://localhost:8080/262';</script>");
    }else{
        response.setCharacterEncoding("utf-8");
        String path=request.getParameter("path");
            …//此处文件下载的相关代码与实例 288 相同
        }
%>
```

（3）为了更好地模拟如何防止网站被盗链下载，这里又单独创建了一个实例，在该实例中只包含一个 test.jsp 文件，在该文件中实现对本实例的下载地址进行盗链，关键代码如下：

```
<a href="http://localhost:8080/262/download.jsp?path=http://localhost:8080/262/测试图片.jpg">下载地址 1  </a>
```

秘笈心法

如果不知道 getHeader()方法中的请求头参数该如何写，可以通过 getHeaderNames()方法返回一个 Enumeration 对象，它包含了 HTTP 请求头部的所有项目名。

实例 290　隐藏文件下载的真实路径

光盘位置：光盘\MR\11\290

中级

实用指数：★★★☆

实例说明

实例 288 和 289 实现了文件下载，在用户单击文件下载超链接时，就会发现文件在服务器的真实路径，这样暴露了服务器的文件路径，可能会造成安全隐患。本实例将介绍在文件下载时隐藏文件在服务器上的真实路径，只显示文件的虚拟路径。运行本实例，如图 11.41 所示，当鼠标经过下载链接处时，在浏览器的状态栏只显示文件在服务器上的虚拟路径，而非真实路径。



图 11.41　隐藏文件下载的真实路径

关键技术

本实例在文件下载超链接处获取文件的虚拟路径主要是通过 JSP 内置对象 request 来完成的。在 JSP 中，可以应用 request 对象获取服务器的相关信息。本实例在构建文件虚拟路径时用到了以下几个方法。

- getScheme()：返回客户端与服务器端通信所用的协议名称。
- getServerName()：返回服务器的主机名。
- getServerPort()：返回服务器的端口号。
- getContextPath()：返回客户端所请求的 Web 应用的 URL 入口。例如，客户端访问 URL 为 http://localhost:8080/test/index.jsp，那么该方法返回值为“/test”。

设计过程

（1）创建 index.jsp 页，为存储在服务器指定文件夹下的文件设置下载超链接，链接形式为文件在服务器中的虚拟路径，链接到 download.jsp 处理页中，关键代码如下：

```
<%
    String path = request.getContextPath();
```

```
    String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";
%>
<A HREF="download.jsp?path=<%=basePath+"测试图片.jpg"%>">
```

（2）创建 download.jsp 文件下载处理页。在该页的请求中获取文件虚拟路径，然后截取文件的名称字符串，再根据文件名称隐藏服务器的真实路径创建 File 对象，实现文件下载功能，关键代码如下：

```
<%
    response.setCharacterEncoding("utf-8");              //设置响应编码
    String path=request.getParameter("path");            //获取文件虚拟路径
    path=new String(path.getBytes("iso-8859-1"));
    String realPath = application.getRealPath("");       //获取 Web 服务器目录
    String fileName = path.substring(path.lastIndexOf("/"));//从虚拟路径中截取文件名
    File file = new File(realPath,fileName);             //创建 File 对象
    InputStream in = new FileInputStream(file);          //创建文件输入流
    OutputStream os = response.getOutputStream();        //获取响应打印输出流
    //设置响应头
    response.addHeader("Content-Disposition", "attachment;filename="+ new String(file.getName().getBytes("gbk"),"iso-8859-1"));
    response.addHeader("Content-Length", file.length() + "");
    response.setContentType("application/octet-stream");
    int data = 0;
    while ((data = in.read()) != -1) {                   //从输入流中读取文件字节
        os.write(data);                                  //输出字节到响应流
    }
    os.close();
    in.close();
%>
```

秘笈心法

在实现文件下载处理时，还是需要获取文件在服务器中的真实路径才可以下载文件，只是本实例没有在超链接处直接暴露文件在服务器的真实路径，获取文件的真实路径是在服务器端处理页中完成的。

实例 291　应用 jspSmartUpload 组件实现文件下载

光盘位置：光盘\MR\11\291

高级

实用指数：★★★☆

实例说明

应用开源的 jspSmartUpload 组件不仅可以实现文件上传功能，而且还可以实现文件的下载功能，应用该组件中提供的方法，可以轻松地实现文件下载功能。本实例将介绍如何应用 jspSmartUpload 组件实现文件下载。运行本实例，如图 11.42 所示，单击下载链接后，浏览器将弹出文件下载的另存为提示框。



图 11.42　应用 jspSmartUpload 组件实现文件下载

关键技术

本实例主要应用 jspSmartUpload 组件中的 SmartUpload 类来实现文件下载，该类是文件上传下载的核心类，应用该类中的相应方法可以实现文件下载功能。SmartUpload 类中用于文件下载的相关方法如表 11.9 所示。

表 11.9　SmartUpload 类的文件下载相关方法

方　法　名	作　　用
initialize(Pagecontext pageContext)	用于执行上传下载的初始化操作，在调用上传下载方法之前，必须先调用该方法进行初始化。该方法用于在 JSP 页中处理文件下载时调用，参数 pageContext 为 JSP 内置对象
downloadFile(String sourceFileName)	文件下载方法，参数为文件的包含路径的全名称
downloadFile(String sourceFileName,String contentType)	文件下载的重载方法，参数 sourceFileName 指下载的源文件名称，contentType 指 MIME 格式的文件类型
downloadFile(String sourceFileName,String contentType,String destFileName)	文件下载的重载方法，参数 sourceFileName 指下载的源文件名称，contentType 指 MIME 格式的文件类型，destFileName 指下载后默认的另存文件名
setContentDisposition(String contentDisposition)	将数据追加到 mime 文件头的 content-disposition 域。jspSmartUpload 组件会在返回下载的信息时自动填写 mime 文件头的 content-disposition 域，如果用户需要添加额外信息，请用此方法

设计过程

（1）创建 index.jsp 页，为存储在服务器指定文件夹下的文件设置下载超链接，链接形式为文件在服务器中的虚拟路径，链接到 download.jsp 处理页中。

（2）创建 download.jsp 文件下载处理页。在该页的请求中获取文件虚拟路径，然后截取文件的名称字符串，再应用 jspSmartUpload 组件实现文件下载功能，关键代码如下：

```
<%@ page import="com.jspsmart.upload.*"  %>
<%
	String from = request.getHeader("referer");			//获取当前请求的上一次访问的地址
	//判断当前请求的上一次访问的地址与指定的地址是否相同
	if ((from == null) || (from.indexOf("localhost:8080/264/") < 0)) {
		out.print("<script>alert('对不起，请您登录正确的网站进行下载！');window.location.href='http://localhost:8080/264';</script>");
	}else{
		response.setCharacterEncoding("utf-8");
		String path=request.getParameter("path");
		path=new String(path.getBytes("iso-8859-1"));
		String fileName = path.substring(path.lastIndexOf("/"));
		SmartUpload su = new SmartUpload();		//新建一个 SmartUpload 对象
		su.initialize(pageContext);			//初始化准备操作
		su.setContentDisposition(null);			//设定 contentDisposition 为 null 以禁止浏览器自动打开文件
		su.downloadFile(fileName);			//下载文件
	}
%>
```

秘笈心法

本实例在实现文件下载时调用 SmartUpload 中的 setContentDisposition()方法，并设置参数为 null，如果 contentDisposition 为 null，则组件将自动添加“"attachment;"”，以表明将下载的文件作为附件，结果是 IE 浏览器将会提示另存文件，而不是自动打开这个文件。当下载文件为 doc 或 pdf 文件时，如果不设置为 null，IE 浏览器会调用程序打开这两种格式的文件。

实例 292 处理 jspSmartUpload 组件下载文件名乱码问题

光盘位置：光盘\MR\11\292

高级

实用指数：★★★★☆

实例说明

在实例 291 中应用 jspSmartUpload 组件实现文件下载时，文件下载的另存文件名存在乱码问题，这是该组件存在的一个问题。本实例将介绍如何处理在应用 jspSmartUpload 组件下载文件时的文件名乱码问题，运行本实例，如图 11.43 所示，在单击文件下载的超链接后，弹出的文件下载另存为提示框中的文件名没有出现乱码问题。



图 11.43　处理 jspSmartUpload 组件实现文件下载时的文件名乱码

关键技术

jspSmartUpload 组件没有对中文进行处理，如果希望该组件支持中文，需要对该组件的源代码进行改进。但是由于其官网 www.jspsmart.com 目前已经停用，所以暂时没有 jspSmartUpload 组件的源码文件，不过可以利用 Java 反编译工具将 jar 包进行反编译，然后就可以修改其源代码。在组件的源码中，只要将文件名参数进行 UTF-8 编码转换后即可解决乱码问题。

设计过程

（1）在 jspSmartUpload 组件的 src 源文件中找到 SmartUpload 类，在该类中编写字符串进行 UTF-8 编码转换的方法 toUtf8String()，参数 str 表示要转换的字符串，具体代码如下：

```
public static String toUtf8String(String str) {
        StringBuffer sb = new StringBuffer();                    //创建 StringBuffer 对象，用于保存编码后的字符串
        for (int i=0;i<str.length();i++) {                      //遍历字符串中每一个字符
                char c = str.charAt(i);
                if (c >= 0 && c <= 255) {                       //字符在 0～255 之间的
                        sb.append(c);
                } else {                                        //字符大于 255 的
                        byte[] bytes;
                        try {
                                bytes = Character.toString(c).getBytes("utf-8");  //将字符转换为字节数组
                        } catch (Exception ex) {
                                ex.printStackTrace();
                                bytes = new byte[0];
                        }
                        for (int j = 0; j < bytes.length; j++) {        //循环字节数组
                                int k = bytes[j];
```

```
                    if (k < 0)
                        k += 256;
                    sb.append("%" + Integer.toHexString(k).toUpperCase());//将字节转换为十六进制字符串
                }
            }
        }
        return sb.toString();
}
```

（2）在 SmartUpload 类的文件下载方法 downloadFile(String s,String s1,String s2,int i)中，在获取文件名处调用步骤（1）编写的字符转换方法，关键代码如下：

```
public void downloadFile(String s, String s1, String s2, int i) throws ServletException, IOException, SmartUploadException{
    if (s == null)
      throw new IllegalArgumentException("File '" + s + "' not found (1040).");
    if (s.equals(""))
      throw new IllegalArgumentException("File '" + s + "' not found (1040).");
    if ((!isVirtual(s)) && (this.m_denyPhysicalPath))
      throw new SecurityException("Physical path is denied (1035).");
    if (isVirtual(s))
      s = this.m_application.getRealPath(s);
    java.io.File file = new java.io.File(s);
    FileInputStream fileinputstream = new FileInputStream(file);
    long l = file.length();
    boolean flag = false;
    int k = 0;
    byte[] abyte0 = new byte[i];
    if (s1 == null) {
      this.m_response.setContentType("application/x-msdownload");
    }
    else if (s1.length() == 0)
      this.m_response.setContentType("application/x-msdownload");
    else
      this.m_response.setContentType(s1);
    this.m_response.setContentLength((int)l);
    this.m_contentDisposition = ((this.m_contentDisposition != null) ? this.m_contentDisposition : "attachment;");
    if (s2 == null) {
     //此处调用字符编码转换的方法
      this.m_response.setHeader("Content-Disposition", this.m_contentDisposition + " filename=" + toUtf8String(getFileName(s)));
    }
    else if (s2.length() == 0)
      this.m_response.setHeader("Content-Disposition", this.m_contentDisposition);
    else
      this.m_response.setHeader("Content-Disposition", this.m_contentDisposition + " filename=" + toUtf8String(s2)); //调用字符编码转换的方法
    while (k < l)
    {
      int j = fileinputstream.read(abyte0, 0, i);
      k += j;
      this.m_response.getOutputStream().write(abyte0, 0, j);
    }
    fileinputstream.close();
  }
```

（3）在操作系统的 cmd.exe 命令行工具中，运行 jar 命令将修改过的 jspSmartUpload 组件源代码打包成 jar 文件，然后将 jar 包复制到项目的 WEB-INF/lib 目录中即可。在命令行中为文件打包的命令如下：

```
jar cvf jspsmartupload_zh_CN.jar com
```

秘笈心法

读者可以将改进过的 jspSmartUpload 组件 jar 包保存起来，在其他项目中应用时，可以直接应用此处理中文的 jar 包。

第12章

文件的批量管理

- ⏭ 文件的批量操作
- ⏭ 文件的压缩与解压缩
- ⏭ 文件的批量上传

12.1　文件的批量操作

在现实应用中，为了实现更高的操作效率，经常需要批量操作文件。本节将介绍一些对文件批量操作比较基础的例子，如对文件的批量重命名、删除文件夹中所有文件、文件搜索等。

实例 293	文件批量重命名 光盘位置：光盘\MR\12\293	中级 实用指数：★★★★

实例说明

Windows 操作系统可以实现重命名文件操作，却不能实现批量重命名。本实例实现了批量重命名功能，可以将一个文件夹内同一类型的文件按照一定的规则批量重命名。用户可以给出重命名模板，程序将根据模板对相应的文件进行重命名。除此之外还可以在重命名模板中添加特殊符号，程序会将这些特殊符号替换成重命名后的文件编号，实例运行效果如图 12.1 所示。



图 12.1　文件批量重命名

关键技术

本实例主要应用了 String 字符串的格式化方法，该方法可以将指定对象按特定的格式生成字符串，本实例格式化的目的是为新文件名称做递增编号的同时保留指定位数的 0 前导数字。例如，3 位编号的 1 应该为 001。字符串类的格式化方法声明如下：

```
public static String format(String format, Object... args)
```

功能：使用指定的格式字符串和参数返回一个格式化字符串。

参数说明

❶ format：格式字符串。

❷ args：格式字符串中由格式说明符引用的参数。

例如，以下代码可以格式化并返回字符串 photo025。

```
String fileName = String.format("photo%04d", 25);
```

设计过程

（1）创建实现 java.io.FileFilter 接口的文件过滤器类 ExtendNameFilter，这个过滤器的任务是只允许获取指

定扩展名的文件对象，它将被应用到遍历文件夹所有文件的 listFiles()方法中，关键代码如下：

```
public class ExtendNameFilter implements FileFilter {
    private String extName;
    public ExtendNameFilter(String extName) {
        this.extName = extName;                                          //保存文件扩展名
    }
    @Override
    public boolean accept(File pathname) {
        if (pathname.getName().toUpperCase().endsWith(extName.toUpperCase()))   //过滤文件扩展名
            return true;
        return false;
    }
}
```

（2）创建工具类 FileUtil，在该类中实现文件的批量重命名。该类包含两个 List 集合类型的成员变量，用于存储文件的旧文件名和重命名之后的新文件名。FileUtil 类的关键代码如下：

```
public class FileUtil {
        private List<String> oldNameList = new ArrayList<String>();          //创建 List 集合，用于保存所有旧文件名
        private List<String> newNameList = new ArrayList<String>();          //创建 List 集合，用于保存所有新文件名
        public List<String> getOldNameList() {
                return oldNameList;
        }
        public List<String> getNewNameList() {
                return newNameList;
        }
        /**
        * 文件批量重命名
        * @param filePath 文件路径
        * @param templet 文件命名模板
        * @param extName 文件扩展名
        */
        public void renameFile(String filePath,String templet ,String extName){
                File dir = new File(filePath);                               //根据文件路径创建 File 对象
                int bi =1;                                                   //起始编号
                int index = templet.indexOf("#");                            //获取第一个“#”的索引
                String code = templet.substring(index);                      //获取模板中数字占位字符串
                templet = templet.replace(code, "%0" + code.length() + "d"); //把模板中数字占位字符串替换为指定格式
                extName = extName.toLowerCase();
                if (extName.indexOf(".") == -1)
                        extName = "." + extName;
                File[] files = dir.listFiles(new ExtendNameFilter(extName)); //获取文件夹中文件列表数组
                for (File file : files) {                                    //变量文件数组
                        String name = templet.format(templet, bi++) + extName;  //格式化每个文件名称
                        oldNameList.add(file.getName());                     //将旧文件名添加到 List 集合中
                        newNameList.add(name);                               //将新文件名添加到 List 集合中
                        File parentFile = file.getParentFile();              //获取文件所在文件夹对象
                        File newFile = new File(parentFile, name);
                        file.renameTo(newFile);                              //文件重命名
                }
        }
}
```

（3）创建 index.jsp 页，在本页中获取表单的请求信息，然后调用 FileUtil 类的 renameFile()方法，实现文件批量重命名，关键代码如下：

```
<%
        request.setCharacterEncoding("UTF-8");                               //设置请求编码
        String submit = request.getParameter("submit");                      //获取“开始重命名”按钮的值
        String filePath = request.getParameter("filePath");                  //获取文件路径
        String tempStr = request.getParameter("tempStr");                    //获取文件名的模板
        String extendName = request.getParameter("extendName");              //获取文件扩展名
        FileUtil fileUtil = new FileUtil();
        if(submit!=null){                                                    //判断是否提交表单
                if(filePath!=null&&!filePath.equals("")&&
                        tempStr!=null&&!tempStr.equals("")&&
                        extendName!=null&&!extendName.equals("")){           //判断是否为空
                        fileUtil.renameFile(filePath,tempStr,extendName);    //调用方法将文件重命名
```

```
        }
    }
%>
```

秘笈心法

文件的名称以不同的后缀来区分其类别，这也称为文件的扩展名，在更改文件名称时不要忘记把扩展名也一同输入，否则如果软件没有特殊处理，可能会把文件变成无类型（即没有扩展名）的文件。

实例 294　**快速批量移动文件**　中级

光盘位置：光盘\MR\12\294　实用指数：★★★☆

实例说明

文件移动是计算机资源管理常用的一个操作，这在操作系统中可以通过文件的剪切与复制来实现，也可以通过鼠标的拖动来实现。但是在 Java 语言的编程实现中，大多是以复制文件到目的地，再删除原有文件来实现的。这对于小文件来说看不出什么弊端，但是如果移动几个大的文件就会看出操作缓慢并且浪费系统资源。本实例将通过 File 类的 API 方法直接实现文件的快速移动，哪怕是上 G 的大文件也不会造成严重延时，程序的运行结果如图 12.2 所示。

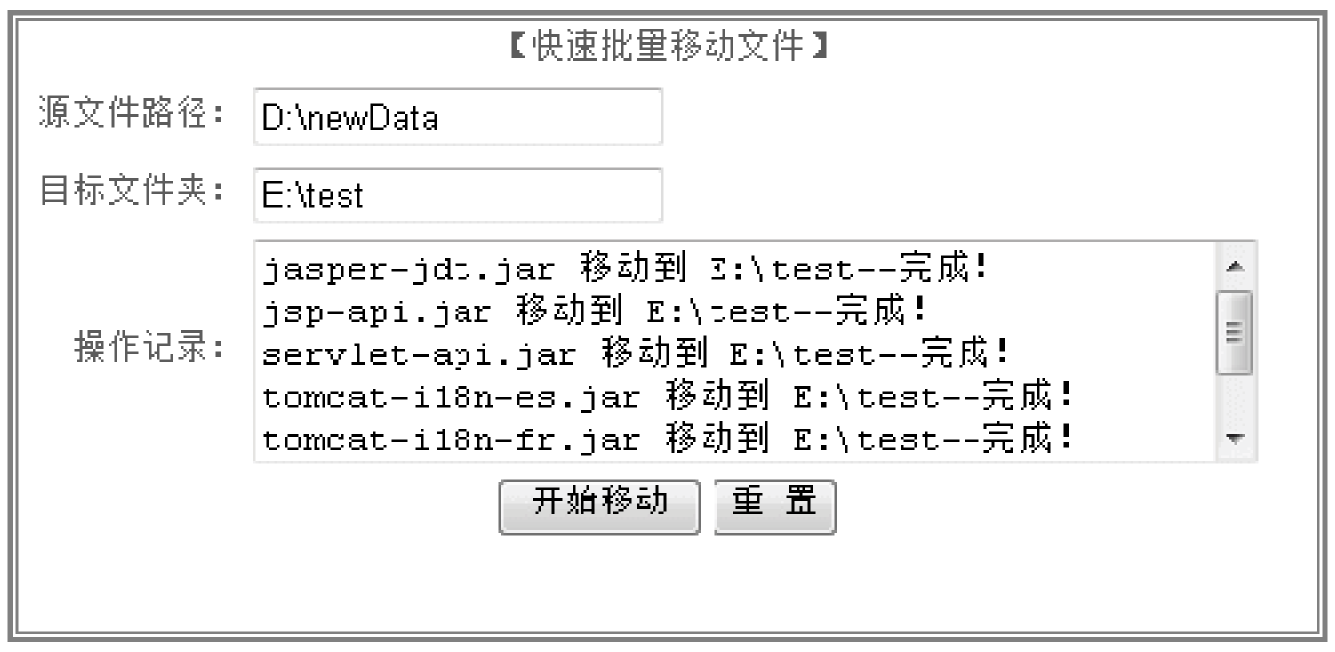


图 12.2　快速批量移动文件

关键技术

File 类位于 Java.io 类包中，它提供了多种获取文件属性的方法，其中 renameTo()方法可以实现文件的重新命名，但是本实例利用该方法将文件路径进行修改，从而实现文件的快速移动。该方法的声明如下：

```
public boolean renameTo(File dest)
```

参数说明

dest：指定文件的新抽象路径名。

对于参数 dest，可以设置不同路径的文件对象，这样就可以实现文件的快速移动。

注意：不同磁盘之间的文件移动会涉及文件的复制与删除操作，速度无法提升，但是这些都由 renameTo()方法自动完成。

设计过程

（1）创建工具类 FileUtil，该类中主要包含一个文件批量移动的方法，并且在类中定义一个用于保存移动操作记录的 List 集合类型的成员变量，关键代码如下：

```
public class FileUtil {
        //创建 List 集合，用于保存文件移动操作记录
        private List<String> workLogList = new ArrayList<String>();
        public List<String> getWorkLogList() {
                return workLogList;
        }
        /**
```

```
     * 快速批量移动文件
     * @param sourcePath 文件源路径
     * @param targetPath 文件目标路径
     * @return 移动成功返回 true，否则返回 false
     */
    public boolean  moveFile(String sourcePath,String targetPath ){
        try {
            File sourceDir = new File(sourcePath);          //根据源路径创建 File 对象
            if(sourceDir.exists()){
                for(File file :sourceDir.listFiles()){      //遍历文件夹下的所有文件
                    File targetFile = new File(targetPath,file.getName());
                    file.renameTo(targetFile);              //根据修改文件路径，实现文件移动
                    workLogList.add(file.getName()+" 移动到 "+targetPath+"--完成！");
                }
            }
            return true;
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }
}
```

（2）创建 index.jsp 页，在本页中获取表单的请求信息，然后调用 FileUtil 类的 moveFile()方法，实现快速批量移动文件，关键代码如下：

```
<%
    request.setCharacterEncoding("UTF-8");                      //设置请求编码
    String submit = request.getParameter("submit");             //获取“开始移动”按钮的值
    String sourcePath = request.getParameter("sourcePath");     //获取源文件夹路径
    String targetPath = request.getParameter("targetPath");     //获取目标文件夹路径
    FileUtil fileUtil = new FileUtil();
    if(submit!=null){                                           //判断是否提交表单
        if(sourcePath!=null&&!sourcePath.equals("")&&
            targetPath!=null&&!targetPath.equals("")){          //判断是否为空
            fileUtil.moveFile(sourcePath,targetPath);           //调用方法移动文件
        }
    }
%>
```

秘笈心法

本实例主要利用 renameTo()方法的特性来实现文件的快速移动，该方法的定义为重命名抽象路径名表示的文件。例如，一个文件为 D:\test\a.txt，当调用 renameTo()方法重命名这个文件时，如重命名为 E:\test\a.txt，那么这个文件会自动被移动到 E:\test 目录下，相当于 Windows 系统下的文件剪切。

实例 295	删除指定磁盘所有.tmp 临时文件 光盘位置：光盘\MR\12\295	高级 实用指数：★★★★

实例说明

在操作系统中有很多程序建立了太多的.tmp 临时文件来为程序提供数据缓冲，这样的文件有的在程序关闭时会自动清理，有的则一直存在。另外，程序在运行期间被非法终止也会导致临时文件的冗余。本实例实现了搜索指定磁盘的.tmp 临时文件并进行清理的功能，实例运行效果如图 12.3 所示。

关键技术

本实例主要应用自定义的递归方法在指定磁盘中搜索所有的.tmp 文件。搜索临时文件时，需要应用 File 类的 listFiles(FileFilter filter)方法过滤以.tmp 为后缀的所有文件，语法格式如下：

```
public File[] listFiles(FileFilter filter)
```

功能：返回抽象路径名数组，这些路径名表示此抽象路径名表示的目录中满足指定过滤器的文件和目录。

参数说明

filter：实现 FileFilter 接口的实例对象，该对象的 accept()方法用于实现文件的过滤。



图 12.3　删除指定磁盘所有.tmp 临时文件

设计过程

（1）创建实现 java.io.FileFilter 接口的 ExtendNameFilter 类，该类的任务就是过滤.tmp 临时文件，关键代码如下：

```
public class ExtendNameFilter implements FileFilter {
     @Override
    public boolean accept(File pathname) {
        if (pathname.getName().endsWith(".tmp") || pathname.isDirectory())
            return true;
        return false;
    }
}
```

（2）创建工具类 FileUtil，在该类中首先创建一个 ExtendNameFilter 过滤器对象和一个保存所有临时文件的 List 集合，然后编写获取所有.tmp 临时文件的递归方法，在递归方法中，将搜索到的所有临时文件都添加到 List 集合中，关键代码如下：

```
public class FileUtil {
     //创建过滤器类，过滤.tmp 文件
     private ExtendNameFilter tmpFilter = new ExtendNameFilter();
     //创建 List 集合，用于保存搜索出的.tmp 临时文件
     private List<File> temFiles = new ArrayList<File>();
     public List<File> getTemFiles() {
          return temFiles;
     }
     /**
     * 递归方法，获取指定磁盘所有临时文件
     * @param driver  磁盘路径
     */
     public void listTempFiles(File driver) {
          File[] files = driver.listFiles(tempFilter);          //获取指定磁盘或文件夹的子列表
          if (files == null)
               return;
          for (File file : files) {                             //遍历文件数组
               if (file.isFile()) {                             //处理文件
               temFiles.add(file);                              //将文件添加到 List 集合
               } else if (file.isDirectory()) {                 //处理文件夹
                    listTempFiles(file);                        //递归方法遍历文件夹
               }
          }
     }
}
```

（3）在 FileUtil 类中编写删除所有临时文件的方法，关键代码如下：

```
public boolean   deleteTmpFile(List<File> files){
     try {
          for (File file:files) {                               //遍历文件数组
```

```
                if(file.exists()){                          //判断文件是否存在
                    file.delete();                          //删除.tmp 临时文件
                }
            }
            return true;
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
}
```

（4）创建 index.jsp 页，在本页中获取表单请求，当单击“搜索”按钮时，调用 FileUtil 类的方法搜索指定磁盘所有临时文件，关键代码如下：

```
<%@ page import="java.util.*" %>
<%@ page import="com.lh.util.*" %>
<%@ page import="java.io.*"%>
<%!List<File> tempFiles = null; //用于保存搜索出的临时文件 %>
<%
    request.setCharacterEncoding("UTF-8");                  //设置请求编码
    String submit = request.getParameter("submit");         //获取“搜索”按钮的值
    String submit1 = request.getParameter("submit1");       //获取“删除”按钮的值
    String driverPath = request.getParameter("driverPath"); //获取文件路径
    FileUtil fileUtil = new FileUtil();
    if(submit!=null){                                       //判断是否提交表单
        if(driverPath!=null&&!driverPath.equals("")){
            fileUtil.listTempFiles(new File(driverPath));   //调用递归方法，获取所有.tmp 文件
            tempFiles = fileUtil.getTemFiles();             //返回保存临时文件的 List 集合
        }
    }
%>
```

（5）当单击“删除”按钮时，调用 FileUtil 类的方法删除所有临时文件，关键代码如下：

```
<%
    if(submit1!=null){                                      //判断是否删除
        boolean res = fileUtil.deleteTmpFile(tempFiles);    //删除所有.tmp 临时文件
        if(res){
            tempFiles = null;                               //清空集合中的元素
            out.println("<script>alert('temp 临时文件删除成功')</script>");
        }
    }
%>
```

秘笈心法

本实例实现的核心之处就是搜索临时文件的递归方法。在递归方法中循环 File 数组时，应用 isFile()方法判断 File 数组中的元素是否为文件，如果是文件则添加到指定集合中；应用 isDirectory()方法判断数组元素是否为文件夹，如果是文件夹，则继续调用本方法进行搜索，直到搜索出所有文件夹中的文件。

实例 296	动态加载磁盘文件 光盘位置：光盘\MR\12\296	初级 实用指数：★★★★

实例说明

在使用图形界面操作系统时，当打开一个文件夹系统时会自动列出该文件夹下的所有文件及子文件夹。本实例实现了类似的功能：首先输入一个文件夹路径，程序会动态列出该文件夹下的所有文件；如果该文件是隐藏文件，就在属性栏中显示“隐藏文件”，实例运行效果如图 12.4 所示。

说明：一个文件的属性还包括可读、可写、可运行等。用户可以根据 File 类中的相关方法实现对其他属性的判断。



图 12.4　实例运行效果

关键技术

Java API 中的 File 类提供了很多与文件属性相关的方法。本实例使用到的方法如表 12.1 所示。

表 12.1　File 类的常用方法

方　法　名	作　　用
getAbsolutePath()	以字符串的形式返回该 File 对象的绝对路径
isFile()	测试该 File 对象是否是一个文件，是则返回 true
isHidden()	测试该 File 对象是否是一个隐藏文件，是则返回 true
listFiles()	如果给定的 File 对象是一个文件夹，则将其转换成 File 数组，数组中包括该文件夹中的文件和子文件夹，否则抛出 NullPointerException

注意： 如果对磁盘使用 isHidden()方法，返回的结果也是 true，如 new File("d:\\"). isHidden()。

设计过程

（1）创建 FileUtil 工具类，在该类中实现根据指定文件夹路径查询所有文件的方法，关键代码如下：

```
public List<File> searchFile(String path) {
	List<File> fileList = new ArrayList<File>();
	File dir = new File(path);
	File[] files = dir.listFiles();
	for (File file:files) {                          //遍历用户选择的文件夹
		if (file.isFile()) {                         //判断是否是一个文件
			fileList.add(file);
		}
		try {
			Thread.sleep(100);                       //线程休眠 0.1 秒实现动态加载
		} catch (InterruptedException e) {
			e.printStackTrace();
		}
	}
	return fileList;
}
```

（2）创建 index.jsp 页，在该页中获取用户输入的文件夹路径，然后调用 FileUtil 类的 searchFile()方法查询所有文件并返回 List 集合，关键代码如下：

```
<%
	List<File> files = null;
	String submit = request.getParameter("submit");     //获取“搜索”按钮的值
	String filePath = request.getParameter("filePath"); //获取文件夹路径
	FileUtil fileUtil = new FileUtil();
	if(submit!=null){                                   //判断是否提交表单
		if(filePath!=null&&!filePath.equals("")){
			files = fileUtil.searchFile(filePath);      //调用方法获取所有文件
		}
	}
%>
```

（3）在 index.jsp 页的表格中，遍历 List 集合输出文件信息，关键代码如下：

```
<%
if(files!=null&&files.size()>0){
        for(File file:files){
%>
<tr>
        <td align="center"><%=file.getName()%></td>
        <td align="center">
        <%if(file.isHidden()){
                out.println("隐藏文件");
        }else{
                out.println(" ");
        } %>
        </td>
</tr>
<% }
}%>
```

秘笈心法

文件的属性包括隐藏、可读、可写、可执行等。在 File 类中，对于上面所说的属性都有相对应的方法进行判断。读者可以认真学习一下，方便以后使用。

实例 297　删除文件夹中所有文件　高级

光盘位置：光盘\MR\12\297　实用指数：★★★

实例说明

删除文件是对文件的常用操作之一，操作系统可以根据用户的选择，删除文件或者文件夹。本实例可以根据用户指定的文件夹删除该文件夹中的所有文件，包括子文件夹和隐藏文件，但是保留用户选择的文件夹，实例运行效果如图 12.5 所示。



图 12.5　删除文件夹中所有文件

关键技术

Java API 中的 File 类提供了很多与文件管理相关的方法。本实例使用到的方法如表 12.2 所示。

表 12.2　File 类的常用方法

方　法　名	作　　用
delete()	如果该 File 对象是一个文件或空文件夹就将其删除
getAbsolutePath()	以字符串的形式返回该 File 对象的绝对路径
isFile()	测试该 File 对象是否是一个文件，是则返回 true
isHidden()	测试该 File 对象是否是一个隐藏文件，是则返回 true
listFiles()	如果给定的 File 对象是一个文件夹，则将其转换成 File 数组，数组中包括该文件夹中的文件和子文件夹，否则抛出 NullPointerException

注意：delete()方法只能用来删除文件和空文件夹，并且被该方法删除的文件不能被恢复。

设计过程

（1）创建工具类 FileUtil，在该类中创建一个 List 集合类型的成员变量 delFiles，用于保存被删除的文件。编写方法 deleteDictionary()来实现删除文件夹及其中内容的功能，参数 rootFile 代表用户想删除的文件夹，关键代码如下：

```
public   void deleteDictionary(File rootFile) {
        if (rootFile.isFile()) {
                delFiles.add(rootFile);                    //将被删除的文件保存到集合中
                rootFile.delete();                         //如果给定的 File 对象是文件就直接删除
        } else {                                           //如果是一个文件夹就将其转换成 File 数组
                File[] files = rootFile.listFiles();
                for (File file : files) {
                        deleteDictionary(file);            //如果不是空文件夹则迭代 deleteDictionary()方法
                }
                rootFile.delete();                         //如果是空文件夹就直接删除
        }
}
```

（2）编写方法 deleteFiles()来实现删除文件夹下所有内容但保留给定文件夹的功能，参数 rootFile 代表用户想删除的文件夹，关键代码如下：

```
public static void deleteFiles(File rootFile) {
        if (rootFile.listFiles().length == 0) {            //如果用户给定的是空文件夹就退出方法
                return;
        } else {
                File[] files = rootFile.listFiles();       //将非空文件夹转换成 File 数组
                for (File file : files) {
                        if (file.isFile()) {
                                delFiles.add(file);        //将被删除的文件保存到集合中
                                file.delete();             //删除指定文件夹下的所有文件
                        } else {
                                if (file.listFiles().length == 0) {
                                        file.delete();     //删除指定文件夹下的所有空文件夹
                                } else {
                                        deleteDictionary(file);   //删除指定文件夹下的所有非空文件夹
                                }
                        }
                }
        }
}
```

说明：上面的方法具有很好的通用性，读者可以将其放在自己的工具包中。

（3）创建 index.jsp 页，在该页中调用 FileUtil 中的方法，删除用户输入的文件夹中的所有文件，关键代码如下：

```
<%
        List<File> delFiles = null;
        request.setCharacterEncoding("UTF-8");             //设置请求编码
        String submit = request.getParameter("submit");    //获取按钮的值
        String filePath = request.getParameter("filePath"); //获取文件路径
        FileUtil fileUtil = new FileUtil();
        if(submit!=null){                                  //判断是否提交表单
                if(filePath!=null&&!filePath.equals("")){
                File file = new File(filePath);
                fileUtil.deleteFiles(file);                //删除文件夹中的所有子文件
                delFiles = fileUtil.getDelFiles();
        }
        }
%>
```

秘笈心法

Java 中将文件和文件夹统一用 File 类管理。文件夹又可以分成空文件夹和非空文件夹。对于不同的类型，

能够使用的方法是不同的。实际应用中，读者一定要注意其中的区别。

实例 298 创建磁盘索引文件

光盘位置：光盘\MR\12\298

中级

实用指数：★★

实例说明

为了提高对磁盘文件的搜索效率，可以创建一个磁盘索引文件，将磁盘中所有文件的路径都保存到该文件中，当需要查找时，在该文件中查找即可，实例运行效果如图 12.6 所示。



图 12.6　创建磁盘索引文件

关键技术

本实例主要是获得磁盘中所有文件的路径，再将它们写入到文本文件中。这些内容可以参考前面的范例。

设计过程

（1）创建工具类 FileUtil，在该类中定义一个 List 集合类型的成员变量，用于保存指定磁盘的所有文件的路径，然后编写获取指定磁盘所有文件路径的方法，将查询出的文件路径添加到 List 集合中，关键代码如下：

```
public  List<String> getFilePath(List<String> list, File rootFile) {
    File[] files = rootFile.listFiles();                //获取磁盘路径下的 File 文件数组
    if (files == null)
        return list;
    for (File file : files) {                           //遍历文件数组
        if (file.isDirectory()) {                       //判断是否为文件夹
            getFilePath(list, file);                    //如果为文件夹，继续调用本方法
        } else {                                        //如果是文件，将文件路径添加到集合
            list.add(file.getAbsolutePath().replace("\\", "/"));
        }
    }
    return list;
}
```

（2）在 FileUtil 类中，编写创建磁盘索引文件的方法 createIndexFile()。在该方法中，将查询出的所有文件的路径写入文本文件中，关键代码如下：

```
public  void createIndexFile(String rootPath,String indexFilePath){
    File rootFile = new File(rootPath);                 //利用用户选择的磁盘创建 File 对象
    StringBuilder sb = new StringBuilder();             //利用 StringBuilder 对象保存写入的索引
    File indexFile = new File(indexFilePath);
    getFilePath(list, rootFile);                        //获得磁盘上所有文件的路径
    for(String pathStr:list){                           //遍历集合，将集合元素添加到 StringBuffer 中
        sb.append(pathStr+"\r\n");
    }
    pathStr = sb.toString();
    FileWriter fileWriter = null;
    try {
        fileWriter = new FileWriter(indexFile);
```

```
            fileWriter.write(sb.toString());                    //向用户选择的文本文件中写入数据
            fileWriter.flush();
        } catch (IOException e) {
            e.printStackTrace();
        }
}
```

（3）创建 index.jsp 页，在该页中获取请求信息，然后调用 FileUtil 类的 createIndexFile()方法创建磁盘索引文件，关键代码如下：

```
<%
    request.setCharacterEncoding("UTF-8");                      //设置请求编码
    String submit = request.getParameter("submit");
    String driverPath = request.getParameter("driverPath");     //获取磁盘路径
    String indexFilePath = request.getParameter("indexPath");   //获取索引文件路径
    FileUtil fileUtil = new FileUtil();
    if(submit!=null){                                           //判断是否提交表单
        if(driverPath!=null&&!driverPath.equals("")&&
            indexFilePath!=null&&!indexFilePath.equals("")){
            fileUtil.createIndexFile(driverPath,indexFilePath); //创建磁盘索引文件
        }
    }
%>
```

秘笈心法

磁盘索引文件就是用一个文件来记录磁盘上所有文件的路径，当需要查询文件时，就不用每次都遍历整个磁盘，提高了效率。创建一个索引文件就是先获得磁盘中所有文件的路径，再写入到索引文件即可。索引文件也可以用属性文件创建，读者可以自行完成。

实例 299　快速全盘查找文件

光盘位置：光盘\MR\12\299

高级

实用指数：★

实例说明

在磁盘上进行文件查找有两种方式：第一种是遍历整个磁盘，获得各个文件的路径。如果路径中含有用户指定的关键字，就保存该路径。最后将结果显示给用户。第二种是使用已经建立好的磁盘索引文件，直接在该文件中进行查找。显然第二种方法速度较快，本实例就是采用这种方法来实现对某一特定文件的查找功能的，实例运行效果如图 12.7 所示。



图 12.7　快速全盘查找文件

关键技术

本实例主要是读取磁盘索引文件，再利用 String 类的 contains()方法找出与用户输入的关键字匹配的结果。该方法的声明如下：

```
contains(CharSequence s)
```

当且仅当此字符串包含指定的 char 值序列时，返回 true。

参数说明

s：要搜索的序列。

注意：*如果文件路径中包含用户输入的关键字，该方法也会返回 true。如果仅希望查找某类具体的文件，可以使用 endWith(String suffix)方法。*

设计过程

（1）创建工具类 FileUtil，在该类中编写查找文件的方法 searchFile()，该方法有两个 String 类型的参数，

分别是用户输入的关键字 keyword 和索引文件的路径 indexFilePath，关键代码如下：

```
public String searchFile(String keyword,String indexFilePath){
	FileReader fileReader = null;
	BufferedReader bufferedReader = null;
	try {
		fileReader = new FileReader(indexFilePath);			//创建 FileReader 对象
		bufferedReader = new BufferedReader(fileReader);
		StringBuilder builder = new StringBuilder();			//利用 StringBuilder 对象保存索引
		String temp = null;
		while ((temp = bufferedReader.readLine()) != null) {		//读入文本文件
			builder.append(temp);
			builder.append("\n");						//在每一行的末尾添加一个分隔符
		}
		String[] rows = builder.toString().split("\n");			//将索引按换行符分割
		StringBuffer sb = new StringBuffer();
		for(String row:rows) {						//遍历读入的文本文件
			if(row.contains(keyword)) {				//判断是否包含指定的关键字
				sb.append(row+"\n");				//返回结果
			}
		}
		return sb.toString();
	} catch (IOException e) {
		e.printStackTrace();
		return "";
	}
}
```

（2）创建 index.jsp 页，在该页中获取表单请求信息，调用 FileUtil 类的 searchFile()方法在索引文件中查找文件，关键代码如下：

```
<%
	String pathStr ="";
	request.setCharacterEncoding("UTF-8");				//设置请求编码
	String submit = request.getParameter("submit");
	String keyword = request.getParameter("keyword");		//获取磁盘路径
	String indexFilePath = request.getParameter("indexPath");	//获取索引文件路径
	FileUtil fileUtil = new FileUtil();
	if(submit!=null){							//判断是否提交表单
		if(keyword!=null&&!keyword.equals("")&&
			indexFilePath!=null&&!indexFilePath.equals("")){
			pathStr = fileUtil.searchFile(keyword,indexFilePath);	//在索引文件中查找文件
		}
	}
%>
```

说明：查询结果在显示给用户之前一定要进行合法性验证，即确定该文件确实存在。这一步请读者在源代码中自行添加。

秘笈心法

使用磁盘索引文件能够提高文件的查询速度，前提是磁盘上的文件不经常发生变化。由于磁盘索引文件不能及时反映出磁盘的状态，所以即使在索引文件中有用户查找的文件存在，在显示结果之前也需要测试一下文件是不是真的存在。

实例 300	获取磁盘所有文本文件 光盘位置：光盘\MR\12\300	初级 实用指数：★

实例说明

本实例和前一个类似，都是利用已经创建好的索引文件进行查找，不过由用户指定要查找的文件变成了查找所有文本文件，实例运行效果如图 12.8 所示。



图 12.8　获取磁盘所有文本文件

关键技术

本实例主要是读取磁盘索引文件，再利用 String 类的 endsWith()方法找出所有以.txt 结尾的路径。该方法的声明如下：

```
endsWith(String suffix)
```

测试该字符串是否以指定的后缀结束。

参数说明

suffix：指定的后缀。

设计过程

（1）创建 FileUtil 工具类，编写根据用户指定的索引文件查找文本文件的 searchFile()方法，该方法返回查找出的所有文本文件路径的字符串，关键代码如下：

```
public String searchFile(String indexFilePath){
    FileReader fileReader = null;
    BufferedReader bufferedReader = null;
    try {
        fileReader = new FileReader(indexFilePath);              //利用用户选择的文件创建 FileReader 对象
        bufferedReader = new BufferedReader(fileReader);
        StringBuilder builder = new StringBuilder();
        String temp = null;
        while ((temp = bufferedReader.readLine()) != null) {     //读入文本文件
            builder.append(temp);
            builder.append("\n");                                //在每一行的末尾添加一个分隔符
        }
        String[] rows = builder.toString().split("\n");          //将索引按换行符分割
        StringBuffer sb = new StringBuffer();
        for(String row:rows) {                                   //遍历读入的文本文件
            if(row.endsWith(".txt")) {                           //判断读入的文本文件是否包含指定的关键字
                sb.append(row+"\n");                             //返回结果
            }
        }
        bufferedReader.close();
        fileReader.close();
        return sb.toString();
    } catch (IOException e) {
        e.printStackTrace();
        return "";
    }
}
```

（2）创建 index.jsp 页，在该页中获取表单请求信息，然后调用 FileUtil 类的 searchFile()方法在索引文件中查找文本文件，关键代码如下：

```
<%
    String pathStr ="";
    request.setCharacterEncoding("UTF-8");                       //设置请求编码
    String submit = request.getParameter("submit");
    String indexFilePath = request.getParameter("indexPath");    //获取索引文件路径
    FileUtil fileUtil = new FileUtil();
    if(submit!=null){                                            //判断是否提交表单
        if(    indexFilePath!=null&&!indexFilePath.equals("")){
            pathStr = fileUtil.searchFile(indexFilePath);        //在索引文件中查找文件
        }
```

```
        }
%>
```

秘笈心法

利用索引文件查找的过程主要分成两步，第一步是读入索引文件，第二步是查找与用户指定类型匹配的文件路径。第二步既可以考虑使用 String 类的 endsWith(String suffix)方法实现，也可以使用正则表达式，这样可以获得更加强大的搜索功能。

实例 301 合并多个 txt 文件

光盘位置：光盘\MR\12\301

初级

实用指数：★★

实例说明

本实例实现的是将任意个 txt 文件合并为一个文件。通过 I/O 流可以实现文件的合并，当然可以对任意格式的文件进行合并。本实例以合并 txt 文件为例介绍如何合并文件，运行结果如图 12.9 所示。

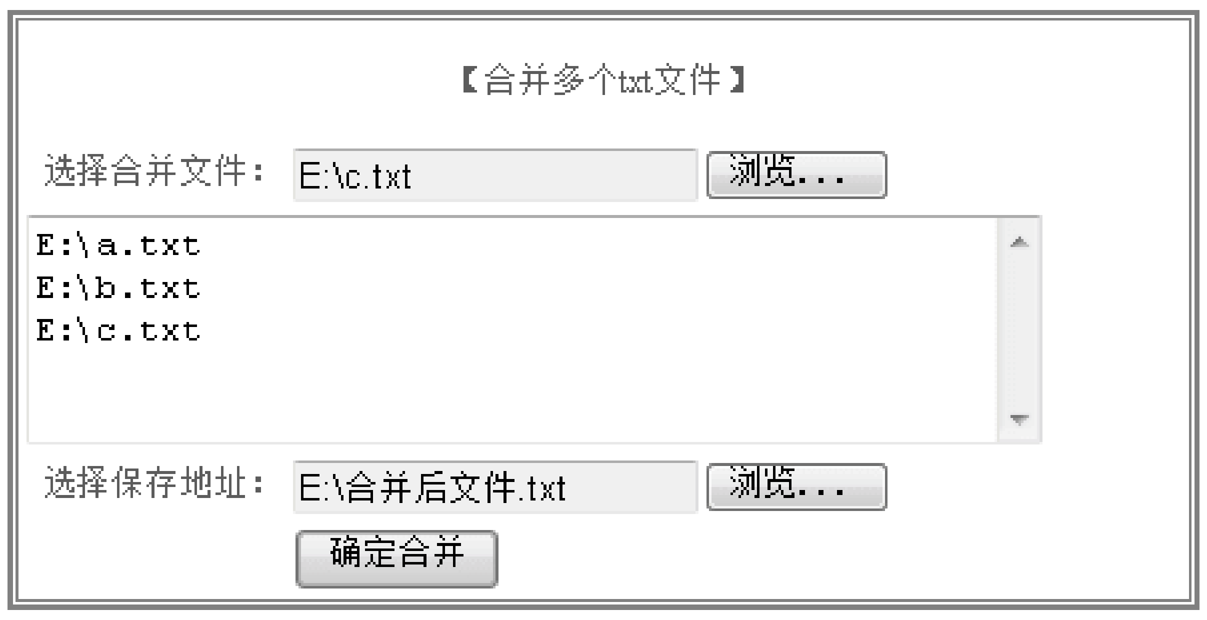


图 12.9　合并多个 txt 文件

关键技术

本实例实现的文件合并主要通过 FileInputStream 类实现读取文件，通过 FileOutputStream 类实现向文件中写入内容。在对文件读取的过程中，本实例应用了一个 FileInputStream 类的一个很重要的方法 available()，获取可读的有效字节数。该方法的语法格式如下：

```
int available()
```

可以通过 FileInputStream 类对象调用该方法。该方法的返回值是，可以从输入流中读取的字节数。

注意：该方法抛出 I/O 异常，在调用该方法时，要通过 try 语句处理异常。

设计过程

（1）创建工具类 FileUtil，在该类中编写合并多个 txt 文件的方法 writeFiles()，关键代码如下：

```
/**
 * 合并多个 txt 文件
 * @param txtPath 用户选择的需要合并的所有文本文件的路径
 * @param savePath 合并后的保存路径
 */
public void writeFiles(String[] txtPath, String savePath) {
        try {
                //根据文件保存地址创建 FileOutputStream 对象
                FileOutputStream fo = new FileOutputStream(savePath, true);
                for (String path:txtPath) {                          //循环遍历文件路径数组
                File file = new File(path);                          //根据文件路径创建 File 对象
                FileInputStream fi1 = new FileInputStream(file);     //创建 FileInputStream 对象
                 byte[] b1 = new byte[fi1.available()];              //从流中获取字节数
                 fi1.read(b1);                                       //读取数据
                 fo.write(b1);                                       //向文件中写数据
                 fi1.close();
```

```
            }
            fo.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
}
```

（2）创建 index.jsp 页，在该页中获取用户选择的所有文件的路径和合并后文件的保存路径，然后调用 FileUtil 类的 writeFiles()方法，实现合并多个 txt 文件，关键代码如下：

```
<%
        request.setCharacterEncoding("UTF-8");                    //设置请求编码
        String submit = request.getParameter("submit");
        String savePath = request.getParameter("savePath");        //获取保存路径
        String txtPath = request.getParameter("files");
        FileUtil fileUtil = new FileUtil();
        if(submit!=null){                                          //判断是否提交表单
                if(txtPath!=null&&!txtPath.equals("")&&savePath!=null&&!savePath.equals("")){
                        String[] files = txtPath.split("\r\n");    //分割所有文件路径的字符串为数组
                        fileUtil.writeFiles(files,savePath);       //调用方法合并多个 txt 文件
                }
        }
%>
```

说明： 由于通过 JavaScript 设置了表单文本域中添加的所有文件路径字符串是以“\r\n”为分隔符的，所以在获取文本域的值时，也应该通过“\r\n”对文件路径字符串进行分解。JavaScript 这一部分的具体代码请参见本书附带的光盘，此处不再具体讲解。

秘笈心法

本实例是以文本文件为例，向大家介绍如何将多个文件合并成一个文件，当然可以合并其他类型的文件，但需要注意合并其他文件时要使用字节流。

实例 302	批量复制指定扩展名的文件 光盘位置：光盘\MR\12\302	中级 实用指数：★★

实例说明

在 Windows 操作系统下可以很轻松地实现复制文件，但是如果要实现批量复制某个类型的文件，就不是很轻松。本实例是一个小工具，通过本实例可以实现将某文件夹下的指定格式的文件复制到相应的文件夹下，运行结果如图 12.10 所示。



图 12.10　批量复制指定扩展名的文件

关键技术

考虑到一个文件夹中可能还包含子文件夹，所以需要应用递归方法循环查找文件夹下所有符合条件的文件，包括文件夹中的子文件夹，一直到查找出所有符合条件的文件为止，递归调用才会结束。

在查询指定扩展名的文件时，需要应用 java.io.FileFilter 接口的实现类对文件扩展名进行过滤，该过滤器的具体用法可参考实例 295，此处不再赘述。

设计过程

（1）创建文件过滤器的类 ExtendNameFilter，该类实现自 java.io.FileFilter 接口，并重写其 accept()方法过滤指定扩展名的文件。ExtendNameFilter 类的具体代码如下：

```
public class ExtendNameFilter implements FileFilter {
  private String extendName;                              //用于过滤的文件扩展名
  public ExtendNameFilter(String extend){
        this.extendName = extend;
  }
  @Override
  public boolean accept(File pathname) {                  //判断文件的扩展名是否符合指定扩展名
      if (pathname.getName().toLowerCase().endsWith(extendName.toLowerCase())||pathname.isDirectory())
            return true;
        return false;
  }
}
```

（2）创建工具类 FileUtil，在该类中定义 getFiles()方法，用于获取某文件夹下的所有文件路径的集合，关键代码如下：

```
/**
 * 递归方法，搜索某个路径下所有指定类型的文件
 * @param path  文件夹路径
 * @param extendName  指定文件的扩展名
 */
public void getFiles(String path,String extendName) {
    File dir = new File(path);                                  //根据文件地址创建 File 对象
    File files[] = dir.listFiles(new ExtendNameFilter(extendName));   //获取文件夹下的文件数组
    for (File file:files ) {                                    //循环遍历数组
        if (file.isDirectory())                                 //判断文件是否是一个目录
            getFiles(file.getAbsolutePath(),extendName);        //如果为文件夹，继续执行本方法
        else {
            allfilePath.add(file.getAbsolutePath());            //将文件路径添加到集合中
        }
    }
}
```

（3）在该 CopyUtil 类中定义复制文件方法 copyFile()，关键代码如下：

```
/**
 * 实现文件复制
 * @param allOldPath  需要复制的所有文件组成的集合
 * @param newPath  文件保存路径
 */
public void   copyFile(List<String> allOldPath, String newPath) {
      for(String oldPath:allOldPath){
            try {
                  int bytesum = 0;
                  int byteread = 0;
                  File oldfile = new File(oldPath);
                  if (oldfile.exists()) {                                   //文件存在时
                        InputStream inStream = new FileInputStream(oldPath);     //读入源文件
                        File newFile = new File(newPath,oldfile.getName());      //在新路径下创建文件
                        newFile.createNewFile();
                        FileOutputStream fs = new FileOutputStream(newFile);     //创建文件输出流
                        byte[] buffer = new byte[1444];
                        while ((byteread = inStream.read(buffer)) != -1) {       //循环读取文件
                              bytesum += byteread;                               //获取文件大小
                              fs.write(buffer, 0, byteread);                     //向文件中写数据
                        }
                        fs.close();
                        inStream.close();
                  }
            } catch (Exception e) {
                  e.printStackTrace();
            }
      }
}
```

（4）创建 index.jsp 页，在该页中获取表单请求的值，然后调用 FileUtil 类的方法实现文件的复制，关键代码如下：

```
<%
      request.setCharacterEncoding("UTF-8");                      //设置请求编码
      String submit = request.getParameter("submit");             //获取按钮的值
```

```
        String filePath = request.getParameter("filePath");              //获取源文件路径
        String savePath = request.getParameter("savePath");              //获取保存路径
        String extendStr = request.getParameter("extendStr");            //获取文件扩展名
        FileUtil fileUtil = new FileUtil();
        if(submit!=null){                                                //判断是否提交表单
                if(filePath!=null&&!filePath.equals("")&&
                        savePath!=null&&!savePath.equals("")&&
                        extendStr!=null&&!extendStr.equals("")){
                        fileUtil.getFiles(filePath,extendStr);           //在文件夹中查找指定扩展名的文件
                        List<String> files =fileUtil.getAllfilePath();
                        fileUtil.copyFile(files,savePath);               //调用方法实现文件复制
                }
        }
%>
```

秘笈心法

在根据文件扩展名实现文件搜索时，尽量应用实现 java.io.FileFilter 接口的文件过滤器类，通过过滤器只需要编写少量的代码就可以实现文件的过滤，这样的代码不仅具有良好的可读性，而且会提高程序的执行效率。

实例 303　将某文件夹中的文件进行分类存储

光盘位置：光盘\MR\12\303

初级　实用指数：★★

实例说明

随着信息技术的高速发展，人们在计算机中存储的文件越来越多，有时由于时间的问题，没有注意文件的存放方式，这样长期下去，计算机中的文件会显得比较凌乱。针对此问题，可以自己开发一个小程序，实现文件的分类存储，将具有相同格式的文件存储在同一个文件夹下，以方便查询，本实例的运行结果如图 12.11 所示。



图 12.11　实现文件的分类存储

关键技术

本实例实现文件分类存储的关键是获取某文件夹下的文件，通过字符串截取的方式提取文件的格式，并根据获取的文件格式创建文件夹。提取文件格式使用的是 String 类的 substring()方法，该方法可在指定的字符串中截取子字符串，语法格式如下：

```
substring(int beginIndex,int endIndex)
```

参数说明

❶ beginIndex：要截取字符串的开始索引位置，包括该索引位置处的字符。

❷ endIndex：要截取字符串的结束索引，不包括该索引位置处的字符。

设计过程

（1）创建工具类 FileUtil，该类定义获取文件夹下所有文件方法 getFiles()，将获取的所有文件路径添加到成员变量 List 集合中，关键代码如下：

```
public void getFiles(String path) {
    File dir = new File(path);                                 //根据文件地址创建 File 对象
    File files[] = dir.listFiles();                            //获取文件夹下的文件数组
    for (File file:files ) {                                   //循环遍历数组
        if (file.isDirectory())                                //判断文件是否是一个目录
            getFiles(file.getAbsolutePath());                  //如果为文件夹，继续执行本方法
        else {
            allfilePath.add(file.getAbsolutePath());           //将文件路径添加到集合中
        }
    }
}
```

（2）在 FileUtil 类中编写新建文件夹方法 createFolder()。该方法有一个 String 类型的参数，用于定义新建文件夹的保存地址，关键代码如下：

```
public void createFolder(String strPath) {
    try {
        File myFilePath = new File(strPath);                    //根据文件地址创建 File 对象
        if (!myFilePath.exists()) {                             //如果指定的 File 对象不存在
            myFilePath.mkdir();                                 //创建目录
        }
    } catch (Exception e) {
        System.out.println("新建文件夹操作出错");
        e.printStackTrace();
    }
}
```

（3）在 FileUtil 类中，编写实现文件分类存储的方法 sortFile()，关键代码如下：

```
public  void sortFile(String filePath) {
    getFiles(filePath);                                         //调用方法，获取用户选择文件夹中所有文件集合
    for(String fileStr:allfilePath ){                           //循环遍历该文件集合
        int index = fileStr.lastIndexOf(".");
        if(index != -1){
            String strN = fileStr.substring(index+1,fileStr.length()); //对文件夹进行截取，获取文件扩展名
            int ind = fileStr.lastIndexOf("\\");
            String strFileName = fileStr.substring(ind, index);
            createFolder(filePath+"\\"+"分类");                    //调用创建文件夹方法，新建文件夹
            createFolder(filePath+"\\"+"分类"+"\\"+strN);
            if(fileStr.endsWith(strN)){
                File oldfile = new File(fileStr);
                File newFile = new File(filePath+"\\"+"分类"+"\\"+strN+"\\"+strFileName+fileStr.substring(index,fileStr.length()));
                oldfile.renameTo(newFile);                      //将文件保存到设置的分类文件夹中
            }
        }
    }
}
```

（4）创建 index.jsp 页，在该页中获取用户输入的文件夹路径，然后调用 FileUtil 类的 sortFile()方法实现文件分类存储，关键代码如下：

```
<%
    request.setCharacterEncoding("UTF-8");                      //设置请求编码
    String submit = request.getParameter("submit");             //获取按钮的值
    String filePath = request.getParameter("filePath");         //获取文件夹路径
    FileUtil fileUtil = new FileUtil();
    if(submit!=null){                                           //判断是否提交表单
        if(filePath!=null&&!filePath.equals("")){
            fileUtil.sortFile(filePath);                        //调用方法实现文件分类存储
            out.println("<script>alert('文件分类成功！');</script>");
        }
    }
%>
```

秘笈心法

在实现将文件夹中的文件复制到分类文件夹中时，并没有通过文件流的方式实现，而使用的是 File 对象的 renameTo()方法，该方法实际上是重命名文件的方法，但是如果修改整个文件的路径名，文件也会被移动到目标路径中。

实例 304　在指定目录下搜索文件

光盘位置：光盘\MR\12\304

高级

实用指数：★★

实例说明

Windows 操作系统下的文件搜索功能大家都很熟悉，通过该功能用户可以在指定的范围内搜索相关的文件。

本实例模拟该功能开发小型的文件搜索工具，支持星号“*”表示任意多个字符，支持“？”表示任意一个字符，运行结果如图 12.12 所示。



图 12.12　在指定目录下搜索文件

关键技术

本实例的实现首先要获取指定目录下的文件数组，再从数组中查询满足条件的文件。获取指定目录下的文件数组，并对文件扩展名进行过滤，可以用 File 类的 listFiles(FileFilter filter)方法，语法格式如下：

```
public File[] listFiles(FileFilter filter)
```

参数说明

filter：文件过滤器。

功能：返回抽象路径名数组，这些路径名表示此抽象路径名所表示目录中满足指定过滤器的文件和目录。除了返回数组中的路径名必须满足过滤器外，此方法的行为与 listFiles()方法相同。

设计过程

（1）编写工具类 FileUtil，在该类中定义 findName()方法，用于查找匹配的文件。如果要查找的文件名与搜索模式匹配，则返回 true，如果不匹配则返回 false，关键代码如下：

```
public static boolean findName(String pattern, String str) {
    int patternLength = pattern.length();                        //获取参数字符串的长度
    int strLength = str.length();
    int strIndex = 0;
    char eachCh;
    for (int i = 0; i < patternLength; i++) {                    //循环字符参数字符串中的每个字符
        eachCh = pattern.charAt(i);                              //获取字符串中每个索引位置的字符
        if (eachCh == '*') {                                     //如果这个字符是一个星号
            while (strIndex < strLength) {
                if (findName(pattern.substring(i + 1), str.substring(strIndex))) { //如果文件名与搜索模型匹配
                    return true;
                }
                strIndex++;
            }
        } else if (eachCh == '?') {                              //如果包含问号
            strIndex++;
            if (strIndex > strLength) {                          //如果 str 中没有字符可以匹配“?”号
                return false;
            }
        } else {                                                 //如果要寻找的是普通文件
             if ((strIndex >= strLength) || (eachCh != str.charAt(strIndex))) { //如果没有查找到匹配的文件
                 return false;
             }
             strIndex++;
        }
    }
    return (strIndex == strLength);
}
```

（2）定义 findFiles()方法，实现文件搜索功能。该方法有 3 个 String 类型的参数，分别用于指定要搜索目录的地址、要搜索文件的名称以及文件的扩展名，关键代码如下：

```
public static List<File> findFiles(String baseDirName, String targetFileName,String extendStr) {
    List<File> fileList = new ArrayList<File>();                 //定义保存返回值的 List 对象
```

```
        File baseDir = new File(baseDirName);                       //根据参数创建 File 对象
        if (!baseDir.exists() || !baseDir.isDirectory()) {          //如果该 File 对象不存在
            return fileList;                                        //返回 List 对象
        }
        String tempName = null;
        File[] files = baseDir.listFiles(new ExtendNameFilter(extendStr));   //获取参数目录下的文件数组，此文件数组是经过指定扩展名过滤的
        for (File file:files) {                                     //循环遍历文件数组
            if (!file.isDirectory()) {                              //如果数组中的文件不是一个目录
                tempName = file.getName();                          //获取该数组的名称
                if (findName(targetFileName, tempName)) {           //调用文件匹配方法
                    fileList.add(file.getAbsoluteFile());           //将指定的文件名添加到集合中
                }
            }
        }
        return fileList;
}
```

（3）编写根据扩展名过滤文件的过滤器类 ExtendNameFilter，该类的具体代码可参考实例 302。

（4）创建 index.jsp 页，在该页中获取用户输入的表单信息，然后调用 FileUtil 类的 findFiles()方法，实现在指定目录下的文件搜索，关键代码如下：

```
<%
    List<File> files = null;
    request.setCharacterEncoding("UTF-8");                      //设置请求编码
    String submit = request.getParameter("submit");             //获取按钮的值
    String filePath = request.getParameter("filePath");         //获取文件路径
    String fileName = request.getParameter("fileName");         //获取文件名关键字
    String extendName = request.getParameter("extendStr");      //文件扩展名
    if(submit!=null){                                           //判断是否提交表单
        if(filePath!=null&&!filePath.equals("")&&
            fileName!=null&&!fileName.equals("")&&
            extendName!=null&&!extendName.equals("")){
            files = FileUtil.findFiles(filePath,fileName,extendName);//调用搜索文件的方法
        }
    }
%>
```

秘笈心法

在应用 List 集合时，尽量采用 List<E>的泛型写法，这种写法可以直接定义 List 集合中的对象类型，方便元素的添加和搜索，简化程序代码。

实例 305	网络文件夹备份 光盘位置：光盘\MR\12\305	中级 实用指数：★★

实例说明

在现代化企业中，公司的电脑是通过组成一个局域网来实现互相交流的。如果想复制另一台电脑中的文件夹该怎么办呢？本实例将讲述如何备份网络文件夹，运行效果如图 12.13 所示。



图 12.13　网络文件夹备份

说明：备份的文件夹地址要使用 URI。如果读者对 URI、URL 和 URN 不熟悉，请参考 Java SE API 中 URI 类的说明。

关键技术

URI 表示一个统一资源标识符的引用。File 类有一个构造方法，可以利用 URI 来获得一个 File，语法格式如下：

```
File(URI uri)
```

参数说明

uri：通过将给定的 file: URI 转换为一个抽象路径名来创建一个新的 File 实例。例如，“d:\\mingrisoft.txt”在转换成 URI 之后就变成了“file:/d:/mingrisoft.txt”。

设计过程

（1）创建工具类 FileUtil，编写复制单个文件的方法 copySingleFile()。参数 source 代表源文件，target 代表宿文件，关键代码如下：

```
public static void copySingleFile(File source, File target) throws IOException {
    FileInputStream input = new FileInputStream(source);          //获得输入流
    FileOutputStream output = new FileOutputStream(target);       //获得输出流
    byte[] b = new byte[1024 * 5];
    int length;
    while ((length = input.read(b)) != -1) {                      //利用循环读取输入流中的全部数据
        output.write(b, 0, length);                               //将输入流中的内容写入到输出流中
    }
    output.flush();                                               //刷新输出流
    output.close();                                               //释放输出流资源
    input.close();                                                //释放输入流资源
}
```

（2）编写工具方法 copyDictionary()，其功能是复制一个文件夹，参数 source 代表源文件夹，target 代表宿文件夹，关键代码如下：

```
public static void copyDictionary(File source, File target) throws IOException {
    File[] files = source.listFiles();                            //将源文件夹转换成 File 数组
    for (File file : files) {
        if (file.isFile()) {                                      //如果是一个文件就调用复制文件的方法
            copySingleFile(file, new File(target.getAbsolutePath() + "/" + file.getName()));
        } else if (file.listFiles().length == 0) {                //如果是一个空文件夹就调用创建文件夹的方法
            new File(target.getAbsolutePath() + "/" + file.getName()).mkdir();
        } else {                                                  //如果是一个非空文件夹就调用自身，进行迭代
            new File(target.getAbsolutePath() + "/" + file.getName()).mkdir();
            copyDictionary(file, new File(target.getAbsolutePath() + "/" + file.getName()));
        }
    }
}
```

技巧： 对于其他来源的文件夹复制，在底层上都可以使用上面的方法实现。区别仅在于文件夹的来源不同。读者可以根据自己的需求，参考 File 类中的构造方法来实现。

（3）创建 index.jsp 页，在该页中获取请求数据，然后调用 FileUtil 类的 copyDictionary()方法，实现文件夹备份，关键代码如下：

```
<%
    request.setCharacterEncoding("UTF-8");                        //设置请求编码
    String submit = request.getParameter("submit");               //获取按钮的值
    String sourcePath = request.getParameter("sourcePath");       //获取源文件夹路径
    String targetPath = request.getParameter("targetPath");       //获取目标文件夹路径
    if(submit!=null){                                             //判断是否提交表单
        if(sourcePath!=null&&!sourcePath.equals("")&&
            targetPath!=null&&!targetPath.equals("")){
            File sourceFolder = new File(new URI(sourcePath));    //以 URI 为参数创建 File 对象
            File targetFolder = new File(targetPath);             //目标文件夹的 File 对象
            FileUtil.copyDictionary(sourceFolder,targetFolder);   //调用方法实现文件夹备份
        }
    }
%>
```

秘笈心法

复制文件夹的过程就是将一个文件夹中的所有文件和文件夹复制到另一个文件夹中。由于 Java 使用 File 类来统一管理文件和文件夹，而对于文件、空文件夹和非空文件夹能够使用的方法并不完全相同，这就需要分类讨论。利用递归的思想，最终可以将任何一个非空文件夹转换成文件和（或）空文件夹的形式，这样就可以统一处理。

12.2 文件的压缩与解压缩

在网络传输文件的过程中，如果文件过大将会影响文件传送的效果和速度，如果将文件通过文件压缩功能将其压缩后再传送，不但可以提高传送速度，还可以节省大量的时间。本节将介绍如何应用程序来实现对文件的压缩与解压缩的功能。

实例 306	压缩所有文本文件 光盘位置：光盘\MR\12\306	初级 实用指数：★★★★

实例说明

在文件传输过程中，如下载文件，用户通常希望在保证文件质量的情况下，文件的体积要尽可能小；对于多个文件可以当成一个文件来传输。利用压缩即可实现上述需求。本实例使用 Java 自带的压缩工具包来实现对于多个文本文件压缩的功能，运行效果如图 12.14 所示。



图 12.14　压缩所有文本文件

关键技术

压缩文件和复制文件类似，只是用另外一种格式来保存输入流。本实例使用到了 ZipOutputStream，它以 ZIP 文件格式写入文件实现输出流过滤器，使用的方法如表 12.3 所示。

表 12.3　ZipOutputStream 常用方法

方　法　名	作　　用
ZipOutputStream(OutputStream out)	创建新的 ZIP 输出流
putNextEntry(ZipEntry e)	开始写入新的 ZIP 文件条目并将流定位到条目数据的开始处

设计过程

（1）创建工具类 FileUtil，实现压缩文件的方法 zipFile()，参数 folder 指明要压缩的文件夹，targetZipFile

指明压缩后生成的文件，关键代码如下：

```
public static void zipFile(File[] files, File targetZipFile) throws IOException {
        File[] txtFiles = folder.listFiles(new ExtendNameFilter());       //获得经过过滤后的所有 txt 文件
        FileOutputStream fos = new FileOutputStream(targetZipFile); //利用给定的 targetZipFile 对象创建文件输出流对象
        ZipOutputStream zos = new ZipOutputStream(fos);               //利用文件输出流创建压缩输出流
        byte[] buffer = new byte[1024];                                //创建写入压缩文件的数组
        for (File file : txtFiles) {                                    //遍历全部文件
                ZipEntry entry = new ZipEntry(file.getName());         //利用每个文件的名字创建 ZipEntry 对象
                FileInputStream fis = new FileInputStream(file);       //利用每个文件创建文件输入流对象
                zos.putNextEntry(entry);                               //在压缩文件中添加一个 ZipEntry 对象
                int read = 0;
                while ((read = fis.read(buffer)) != -1) {
                        zos.write(buffer, 0, read);                    //将输入写入到压缩文件
                }
                zos.closeEntry();                                      //关闭 ZipEntry
                fis.close();                                           //释放资源
        }
        zos.close();
        fos.close();
}
```

注意： 对于写入到 ZIP 文件的每个文件，都要创建一个 ZipEntry 对象来区别，不同文件的 ZipEntry 要不同。通常可以考虑使用文件名来构造 ZipEntry 对象。

（2）创建 index.jsp，在该页中获取用户输入的文件夹路径，然后调用 FileUtil 类的 zipFile()方法，实现对文件夹中所有 txt 文件的压缩，关键代码如下：

```
<%
        request.setCharacterEncoding("UTF-8");                         //设置请求编码
        String submit = request.getParameter("submit");                //获取按钮的值
        String filePath = request.getParameter("filePath");            //获取文件夹路径
        FileUtil fileUtil = new FileUtil();
        if(submit!=null){                                              //判断是否提交表单
                if(filePath!=null&&!filePath.equals("")){
                        File folder = new File(filePath);              //根据源文件夹创建 File 对象
                        //创建新的 zip 文件，文件名为源文件夹的名称与_txtFile 的组合
                        File targetZipFile = new File(folder.getParent(), folder.getName()+"_txtFile.zip");
                        if(!targetZipFile.exists()){
                                targetZipFile.createNewFile();
                        }
                        FileUtil.zipFile(folder,targetZipFile);        //调用方法实现文件压缩
                }
        }
%>
```

秘笈心法

文件的压缩过程不过是把文件的输入流用另外一种格式来保存而已。Java 中每个被压缩的文件都要使用 ZipEntry 来区别，最后将文件的数据写入 ZIP 文件中即可。

实例 307　压缩包解压到指定文件夹

光盘位置：光盘\MR\12\307

中级　实用指数：★★

实例说明

在获得一个以 ZIP 格式压缩的文件后，需要将其进行解压缩，还原成压缩前的文件。本实例使用 Java 自带的压缩工具包来实现解压缩文件到指定文件夹的功能，运行效果如图 12.15 所示。

说明： 用户需要选择压缩文件和解压到哪个文件夹。在解压缩完成后显示解压了的文件。压缩包中的文本文件不要放在文件夹中，否则会出现异常。



图 12.15　压缩包解压到指定文件夹

关键技术

ZipFile 是用来从 ZIP 文件中读取 ZipEntry 的类。本实例使用的方法如表 12.4 所示。

表 12.4　ZipFile 类的常用方法

方　法　名	作　　用
close()	关闭 ZIP 文件
entries()	返回 ZIP 文件条目的枚举
getInputStream(ZipEntry entry)	返回输入流以读取指定 ZIP 文件条目的内容

设计过程

（1）创建工具类 FileUtil，在该类中编写解压缩 ZIP 文件的方法 unZipFiles()，参数 zipFilePath 表示用户选择的 ZIP 文件的路径字符串，参数 targetFolder 表示文件的解压路径，关键代码如下：

```
public void unZipFiles(String zipFilePath,String targetFolder){
	ZipFile zf = null;
	try {
		zf = new ZipFile(zipFilePath);				//利用用户选择的 ZIP 文件创建 ZipFile 对象
		Enumeration e = zf.entries();				//创建枚举变量
		while (e.hasMoreElements()) {				//遍历枚举变量
			ZipEntry entry = (ZipEntry) e.nextElement();	//获得 ZipEntry 对象
			if (!entry.getName().endsWith(".txt")) {		//如果不是文本文件就不进行解压缩
				continue;
			}
			//利用用户选择的文件夹和 ZipEntry 对象名称创建解压后的文件
			File currentFile = new File(targetFolder + File.separator + entry.getName());
			FileOutputStream out = new FileOutputStream(currentFile);
			InputStream in = zf.getInputStream(entry);	//利用获得的 ZipEntry 对象的输入流
			int buffer = 0;
			while ((buffer = in.read()) != -1) {		//将输入流写入到本地文件
				out.write(buffer);
			}
			in.close();
			out.close();
		}
		zf.close();
	} catch (Exception e) {
		e.printStackTrace();
	}
}
```

注意：对于读到的每一个 ZipEntry，都要进行一次写入数据的处理，这样才能还原成原来的文件。

（2）创建 index.jsp 页，在该页中获取用户提交表单的 ZIP 文件路径和要解压的文件夹目录，然后调用 File Util 类的 unZipFiles()方法，实现 ZIP 文件的解压，关键代码如下：

```
<%
	request.setCharacterEncoding("UTF-8");			//设置请求编码
	String submit = request.getParameter("submit");		//获取按钮的值
	String zipFilePath = request.getParameter("zipFile");	//获取压缩文件路径
	String targetPath = request.getParameter("targetPath");	//获取目标文件夹路径
```

```
        FileUtil fileUtil = new FileUtil();
        if(submit!=null){                                      //判断是否提交表单
            if(zipFilePath!=null&&!zipFilePath.equals("")&&
                targetPath!=null&&!targetPath.equals("")){
                fileUtil.unZipFiles(zipFilePath,targetPath);   //调用方法将 ZIP 文件解压
            }
        }
%>
```

秘笈心法

解压缩文件首先要把 ZIP 文件转换成一个 ZipFile 对象，再利用 ZipEntry 分割各个被压缩的文件，将每个 ZipEntry 还原成一个文件即可。

实例 308　压缩所有子文件夹

光盘位置：光盘\MR\12\308

高级

实用指数：★★

实例说明

在压缩文件时，通常情况下一个文件夹都会有若干个子文件夹。此时该怎样处理呢？本实例将展示如何压缩包含子文件夹的文件夹，运行效果如图 12.16 所示。



图 12.16　压缩所有子文件夹

说明： 用户选择需要压缩的文件夹。在压缩完成后，表格中显示压缩的文件。压缩文件与用户选择的文件夹同名，并且位于同一文件夹中。

关键技术

本实例在实现压缩所有子文件夹时，由于在压缩包中是要包含文件的路径的，因此需要获取到文件夹下所有子文件的路径，包括子文件夹下的文件路径。针对此问题，可以应用递归的方法来实现。递归方法在前面的实例中已经多次用到，主要是根据 File 类的 isDirectory()来判断当前文件是否为文件夹，如果是文件夹则会继续调用本方法来查找该文件夹下的文件；否则将文件路径添加到 List 集合中。

设计过程

（1）创建工具类 FileUtil，在该类中定义一个 List 集合类型的全局变量 filePaths，用于保存搜索指定目录中所有子文件的路径字符串。编写获取指定目录的所有文件路径的方法 getFiles(String path)，参数 path 表示要搜索的目录，将搜索出的所有文件路径添加到 filePaths 集合中，关键代码如下：

```
public void getFiles(String path) {
    File dir = new File(path);                          //根据文件地址创建 File 对象
    File files[] = dir.listFiles();                     //获取文件夹下的文件数组
    for (File file:files ) {                            //循环遍历数组
        if (file.isDirectory())                         //判断文件是否是一个目录
            getFiles(file.getAbsolutePath());           //如果为文件夹，继续执行本方法
        else {
            filePaths.add(file.getAbsolutePath());      //将文件路径添加到集合中
        }
    }
}
```

（2）编写实现压缩功能的方法 zipFile()。参数 sourceFolder 指明要压缩的文件夹路径，targetZipFile 指明生成的压缩文件的保存位置，base 指明压缩文件夹的基本路径（如果要压缩的文件夹是“d:\资料”，那么根路径就是“d:\资料”），关键代码如下：

```
public  void zipFile(String sourceFolder, File targetZipFile, String base)
        throws IOException {
        getFiles(sourceFolder);                                     //调用方法获取指定目录下的所有文件路径

        FileOutputStream fos = new FileOutputStream(targetZipFile); //根据给定的 targetZipFile 创建文件输出流对象
        ZipOutputStream zos = new ZipOutputStream(fos);             //利用文件输出流对象创建 Zip 输出流对象
        byte[] buffer = new byte[1024];
        for (String filePath : filePaths) {                         //遍历所有要压缩文件的路径
                File currentFile = new File(filePath);
                ZipEntry entry = new ZipEntry(filePath.substring(base.length() + 1, filePath.length()));//利用要压缩文件的相对路径创建 ZipEntry 对象
                FileInputStream fis = new FileInputStream(currentFile);
                zos.putNextEntry(entry);
                int read = 0;
                while ((read = fis.read(buffer)) != -1) {           //将数据写入到 Zip 输出流中
                        zos.write(buffer, 0, read);
                }
                zos.closeEntry();                                   //关闭 ZipEntry 对象
                fis.close();
        }
        zos.close();                                                //释放资源
        fos.close();
}
```

（3）创建 index.jsp 页，在该页中获取用户输入的要压缩的文件夹路径，然后调用 FileUtil 类的 zipFile()方法实现压缩文件夹中的所有子文件夹，关键代码如下：

```
<%
        request.setCharacterEncoding("UTF-8");                      //设置请求编码
        String submit = request.getParameter("submit");             //获取按钮的值
        String filePath = request.getParameter("filePath");         //获取源文件夹路径
        FileUtil fileUtil = new FileUtil();
        if(submit!=null){                                           //判断是否提交表单
                if(filePath!=null&&!filePath.equals("")){
                        File folder = new File(filePath);
                        File zipFile = new File(folder.getParent(),folder.getName()+"_File.zip");
                        zipFile.createNewFile();
                        fileUtil.zipFile(filePath,zipFile,forder.getParent());  //调用方法实现文件压缩
                }
        }
%>
```

秘笈心法

压缩包含子文件夹的文件夹的想法和压缩全是文件的文件夹类似，区别在于如何找出包含子文件夹的文件夹的所有文件，并且构造 ZipEntry 时不会出现重名现象。本实例采用获得要压缩文件夹中所有文件的相对路径来创新地解决了这个问题。

实例 309 深层文件夹压缩包的释放

光盘位置：光盘\MR\12\309

高级

实用指数：★★

实例说明

本实例在实例 308 的基础上，实现了释放含有子文件夹的压缩文件的功能。运行本实例，用户需要选择压缩文件，在解压缩完成后显示解压了的文件，解压缩后生成的文件夹与用户选择的 ZIP 文件在同一个目录下，如图 12.17 所示。



图 12.17　深层文件夹压缩包的释放

关键技术

本实例在解压文件时，一定要判断文件的解压目录是否存在、判断目录或文件是否存在，这时应用的是 File 类的 exists()方法。如果目录不存在，需要应用 File 类的 mkdirs()方法来创建目录，语法格式如下：

```
public boolean exists()
```

功能：测试此抽象路径名表示的文件或目录是否存在。当且仅当此抽象路径名表示的文件或目录存在时，返回 true；否则返回 false。

```
public boolean mkdir()
```

功能：创建此抽象路径名指定的目录。当且仅当已创建目录时，返回 true；否则返回 false。

```
public boolean mkdirs()
```

功能：创建此抽象路径名指定的目录，包括所有必需但不存在的父目录。注意，此操作失败时也可能已经成功地创建了一部分必需的父目录。当且仅当已创建目录时，返回 true；否则返回 false。

设计过程

（1）创建 FileUtil 工具类，编写实现解压缩文件的方法 unZipFile()，参数 zipFile 指明要解压缩的 ZIP 文件，targetFile 指明要解压到的文件夹，list 是解压缩后生成的文件的路径，关键代码如下：

```
public static void unZipFile(File zipFile, File targetFile, List<String> list)
        throws IOException {
        ZipInputStream in = new ZipInputStream(new FileInputStream(zipFile));    //利用用户选择的 ZIP 文件创建 ZipInputStream 对象
        ZipEntry entry;
        while ((entry = in.getNextEntry()) != null) {                          //遍历所有 ZipEntry 对象
                if (entry.isDirectory()) {                                      //如果 ZipEntry 对象是一个文件夹就进行迭代
                        new File(targetFile+entry.getName()).mkdirs();
                } else {                                                        //如果是文件就将它写入到指定的文件夹中
                        File file = new File(targetFile , entry.getName());
                        File folder = new File(file.getParent());               //根据文件父目录创建 File 对象
                        if(!folder.exists())                                    //如果目录不存在，则创建目录
                                folder.mkdirs();
                        if(!file.exists())                                      //如果文件不存在，则创建文件
                                file.createNewFile();
                        list.add(file.getName());                               //将新生成的文件的路径添加到集合中
                        FileOutputStream out = new FileOutputStream(file);
                        int b;
                        while ((b = in.read()) != -1) {                         //写入数据
                                out.write(b);
                        }
                        out.close();                                            //释放资源
                }
                entry.clone();
        }
        in.close();
}
```

（2）创建 index.jsp 页，在该页中获取用户选择的 ZIP 文件和输入的解压缩地址，调用 FileUtil 类的 unZipFile()方法，实现解压缩 ZIP 文件，关键代码如下：

```
<%
    List<String> allFilePath = null;
    request.setCharacterEncoding("UTF-8");                  //设置请求编码
    String submit = request.getParameter("submit");         //获取按钮的值
    String zipFilePath = request.getParameter("zipFile");    //获取压缩文件的路径
    String targetPath = request.getParameter("targetPath");  //获取目标文件夹路径
    if(submit!=null){                                        //判断是否单击按钮提交表单
        if(zipFilePath!=null&&!zipFilePath.equals("")&&
            targetPath!=null&&!targetPath.equals("")){       //验证表单的值是否为空
            File zipFile = new File(zipFilePath);            //创建 ZIP 文件的 File 对象
            File targetFile = new File(targetPath);          //创建解压缩目录的 File 对象
            if(!targetFile.exists())                         //如果用户输入的目录不存在，则创建该目录
                targetFile.mkdirs();
            allFilePath = new ArrayList<String>();
            FileUtil.unZipFile(zipFile,targetFile,allFilePath);
        }
    }
%>
```

秘笈心法

解压缩包含子文件夹的文件夹的想法和解压缩全是文件的文件夹类似，区别在于如何找出包含子文件夹的文件夹的所有文件，并且构造 ZipEntry 时不会出现重名现象。

实例 310 解决压缩包中文乱码

光盘位置：光盘\MR\12\310

高级

实用指数：★★★

实例说明

在使用 Java 自带的 ZIP 工具类时，会出现中文乱码的问题。为了完善工具箱，本实例使用 Apache 的 ant 组件来解决压缩包中文乱码的问题，运行效果如图 12.18 和图 12.19 所示。



图 12.18　解决压缩包中文乱码



图 12.19　没有乱码的压缩包（左侧）和有乱码的压缩包（右侧）

关键技术

Apache 的 Ant 包提供了对压缩文件功能的支持，该组件包是开源的，包文件可以到 Apache 的官方网站

（http://www.apache.org）进行下载。在 Ant 包中的 org.apache.tools.zip.ZipOutputStream 类实现了 java.util.ZipFile 的功能，并且还对它进行了扩展。本实例使用到的方法如表 12.5 所示。

表 12.5　ZipOutputStream 类的常用方法

方　法　名	作　　用
ZipOutputStream(java.io.OutputStream out)	利用底层的输出流对象创建新的 ZIP 输出流对象
close()	关闭输出流并释放相关资源
closeEntry()	将所有必需的数据写入到当前 entry 中
putNextEntry(ZipEntry ze)	开始写入下一个 ZipEntry 对象
write(byte[] b, int offset, int length)	写入数据

设计过程

（1）创建工具类 FileUtil，编写实现压缩功能的方法 zipFile()。参数 sourceFolder 指明要压缩的文件夹路径，targetZipFile 指明生成的压缩文件的保存位置，base 指明压缩文件夹的基路径（如果要压缩的文件夹是“d:\资料”，那么根路径就是“d:\资料”），关键代码如下：

```
public   void zipFile(String sourceFolder, File targetZipFile, String base)
        throws IOException {
        getFiles(sourceFolder);                                        //调用方法获取指定目录下的所有文件路径

        FileOutputStream fos = new FileOutputStream(targetZipFile); //根据给定的 targetZipFile 创建文件输出流对象
        ZipOutputStream zos = new ZipOutputStream(fos);             //利用文件输出流对象创建 Zip 输出流对象
        byte[] buffer = new byte[1024];
        for (String filePath : filePaths) {                         //遍历所有要压缩文件的路径
                File currentFile = new File(filePath);
                ZipEntry entry = new ZipEntry(filePath.substring(base.length() + 1, filePath.length()));//利用要压缩文件的相对路径创建 ZipEntry 对象

                FileInputStream fis = new FileInputStream(currentFile);
                zos.putNextEntry(entry);
                int read = 0;
                while ((read = fis.read(buffer)) != -1) {           //将数据写入到 Zip 输出流中
                        zos.write(buffer, 0, read);
                }
                zos.closeEntry();                                   //关闭 ZipEntry 对象
                fis.close();
        }
        zos.close();                                                //释放资源
        fos.close();
}
```

（2）创建 index.jsp 页，在该页中获取用户输入的要压缩的文件夹路径，然后调用 FileUtil 类的 zipFile()方法实现压缩文件夹中的所有子文件夹，关键代码如下：

```
<%
        request.setCharacterEncoding("UTF-8");                      //设置请求编码
        String submit = request.getParameter("submit");             //获取按钮的值
        String filePath = request.getParameter("filePath");         //获取文件夹路径
        FileUtil fileUtil = new FileUtil();
        if(submit!=null){                                           //判断是否提交表单
                if(filePath!=null&&!filePath.equals("")){
                        File folder = new File(filePath);
                        File zipFile = new File(folder.getParent(),folder.getName()+"_File.zip");
                        zipFile.createNewFile();
                        fileUtil.zipFile(filePath,zipFile,forder.getParent());//调用方法实现文件压缩
                }
        }
 %>
```

秘笈心法

使用 Java 自带的压缩工具类压缩文件名中有中文的文件（夹）时会出现乱码的问题，此时可以考虑使用

Apache 的 Ant 包，它增加了对中文的支持。

实例 311　Apache 实现文件解压缩

光盘位置：光盘\MR\12\311

高级

实用指数：★★★

实例说明

在获得一个以 ZIP 格式压缩的文件后，需要将其进行解压缩，还原成压缩前的文件。然而当被压缩的文件名中有中文时，Java 自带的压缩工具类会出现 java.lang.IllegalArgumentException 异常。因此本实例使用 Ant 实现文件解压缩功能，实例的运行效果如图 12.20 所示。



图 12.20　Apache 实现文件解压缩

关键技术

Apache 的 Ant 包提供了对压缩文件功能的支持。它的 org.apache.tools.zip.ZipFile 类可以作为 java.util.ZipFile 的替代品。除了 UTF-8，该类对于文件名编码方式还提供了其他支持，这样就可以避免中文乱码的问题。本实例使用到的方法如表 12.6 所示。

表 12.6　ZipFile 类的常用方法

方　法　名	作　　用
ZipFile(java.io.File f)	利用给定的文件对象创建 ZipFile 对象
close()	关闭压缩文档
getEntries()	获得所有的 ZipEntry 对象
getInputStream(ZipEntry ze)	返回读取给定 ZipEntry 对象的输入流

设计过程

（1）创建 FileUtil 工具类，编写实现解压缩文件的方法 unZipFile()，参数 zipFile 指明要解压缩的 ZIP 文件，targetFile 指明要解压到的文件夹，list 是解压缩后生成的文件的路径，关键代码如下：

```
public static void unZipFile(File zipFile, File targetFile, List<String> list)
        throws IOException {
        ZipFile zf = new ZipFile(zipFile);                              //创建 ZipFile 对象
        Enumeration e = zf.getEntries();
        while (e.hasMoreElements()) {                                   //遍历所有 ZipEntry 对象
                ZipEntry entry = (ZipEntry) e.nextElement();
                if (entry.isDirectory()) {                              //如果 ZipEntry 对象是一个文件夹就创建这个文件夹
                        new File(targetFile + entry.getName()).mkdir();
                } else {
                        File file = new File(targetFile + entry.getName());     //获得被压缩的文件
                        File temp = new File(file.getParent());                 //获得被压缩文件的父文件夹
```

```
                    if(!temp.exists())                          //如果父文件夹不存在，则创建
                        temp.mkdirs();
                    if(!file.exists())                          //如果文件不存在，则创建
                        file.createNewFile();
                    list.add(file.getName());                   //将文件的路径保存到列表中
                    FileOutputStream out = new FileOutputStream(file);
                    InputStream in = zf.getInputStream(entry);
                    int b;
                    while ((b = in.read()) != -1) {             //将数据写入到创建好的文件
                        out.write(b);
                    }
                    out.close();                                //释放资源
                }
            }
            zf.close();                                         //释放资源
}
```

（2）创建 index.jsp 页，在该页中获取用户选择的 ZIP 文件和输入的解压缩地址，调用 FileUtil 类的 unZipFile()方法，实现解压缩 ZIP 文件，关键代码如下：

```
<%
        List<String> allFilePath = null;
        request.setCharacterEncoding("UTF-8");                  //设置请求编码
        String submit = request.getParameter("submit");         //获取按钮的值
        String zipFilePath = request.getParameter("zipFile");   //获取压缩文件的路径
        String targetPath = request.getParameter("targetPath"); //获取目标文件夹路径
        if(submit!=null){                                       //判断是否单击按钮提交表单
            if(zipFilePath!=null&&!zipFilePath.equals("")&&
                targetPath!=null&&!targetPath.equals("")){      //验证表单的值是否为空
                File zipFile = new File(zipFilePath);           //创建 ZIP 文件的 File 对象
                File targetFile = new File(targetPath);         //创建解压缩目录的 File 对象
                if(!targetFile.exists())                        //如果用户输入的目录不存在，则创建该目录
                    targetFile.mkdirs();
                allFilePath = new ArrayList<String>();
                FileUtil.unZipFile(zipFile,targetFile,allFilePath);
            }
        }
        %>
```

秘笈心法

解压缩文件可以把 ZIP 文件转换成一个 ZipFile 对象，再利用 ZipEntry 分割各个被压缩的文件，将每个 ZipEntry 还原成一个文件即可。

实例 312　解压缩 Java 对象

光盘位置：光盘\MR\12\312

高级

实用指数：★

实例说明

对于一个已经被序列化的对象，如果要将其还原该怎么办呢？本实例在实例 311 的基础上实现对序列化文件的解压缩操作和反序列化操作。运行本实例，用户需要选择序列化文件的压缩文件，单击“反序列化”按钮即可实现 Java 对象的反序列化，运行效果如图 12.21 所示。



图 12.21　解压缩文件实现 Java 对象反序列化

关键技术

ObjectInputStream 类位于 java.io 包中，用于恢复那些以前序列化的对象。本实例使用的方法如表 12.7 所示。

表 12.7　ObjectInputStream 类的常用方法

方　法　名	作　　用
ObjectInputStream(InputStream in)	创建从指定 InputStream 读取的 ObjectInputStream
readObject()	从 ObjectInputStream 读取对象

设计过程

（1）创建 FileUtil 工具类，编写方法 unzipSerializationObject()实现解压缩文件和反序列化功能，参数 file 指明要解压缩的文件，关键代码如下：

```
public static void unzipSerializationObject(File file)
        throws IOException, ClassNotFoundException {
        ZipFile zipFile = new ZipFile(file);                        //创建 ZipFile 对象
        File currentFile = null;
        Enumeration e = zipFile.entries();
        while (e.hasMoreElements()) {
                ZipEntry entry = (ZipEntry) e.nextElement();
                if (!entry.getName().endsWith(".dat")) {            //遇到后缀名是.dat 的文件就进行解压缩
                        continue;
                }
                currentFile = new File(file.getParent(),entry.getName());
                FileOutputStream out = new FileOutputStream(currentFile);
                InputStream in = zipFile.getInputStream(entry);
                int buffer = 0;
                while ((buffer = in.read()) != -1) {                //写入文件
                        out.write(buffer);
                }
                in.close();                                         //释放资源
                out.close();
        }
        FileInputStream in = new FileInputStream(currentFile);
        ObjectInputStream ois = new ObjectInputStream(in);          //读入解压缩后的文件
        currentFile.delete();                                       //删除解压缩产生的文件
}
```

（2）创建 index.jsp 页，在该页中调用 FileUtil 类的方法，实现解压缩文件和反序列化 Java 对象，关键代码如下：

```
<%
        request.setCharacterEncoding("UTF-8");                      //设置请求编码
        String submit = request.getParameter("submit");             //获取按钮的值
        String zipFilePath = request.getParameter("zipFile");       //获取用户选择的文件路径
        if(submit!=null){                                           //判断是否提交表单
                if(zipFilePath!=null&&!zipFilePath.equals("")){
                        File zipFile = new File(zipFilePath);
                        FileUtil.unzipSerializationObject(zipFile); //实现文件解压缩并反序列化 Java 对象
                }
        }
%>
```

秘笈心法

当使用序列化来保存对象后，如果需要再次将序列化文件还原成原来的对象，就要进行反序列化。反序列化就是打开字节流并且重构对象，此时新的对象和序列化时的对象是一样的。如果读者在前面的实例中改变了对象的状态，那么在本实例中反序列化生成的对象也保存了前面的操作。

实例 313　文件压缩为 RAR 文档

光盘位置：光盘\MR\12\313

高级

实用指数：★★

实例说明

文件压缩是对数据的一种紧凑存储格式，通过压缩能够使文件更小，占用更少的磁盘空间，同时也可以减少网络传输的时间。本程序实现文件到 RAR 文档的压缩，运行效果如图 12.22 所示。



图 12.22　文件压缩为 RAR 文档

关键技术

Runtime 类是每个 Java 程序都内置的一个运行时对象。通过这个对象可以执行外部命令，这样就可以执行 RAR 的压缩、解压缩、添加注释等各种命令。但是这个类不能直接创建对象，需要使用静态方法来获取实例对象并且调用对象的方法来执行外部命令。方法的声明如下：

（1）获取 Runtime 实例对象

```
public static Runtime getRuntime()
```

返回与当前 Java 应用程序相关的运行时对象。Runtime 类的大多数方法是实例方法，并且必须根据当前的运行时对象对其进行调用。

（2）执行外部命令

```
public Process exec(String command) throws IOException
```

在单独的进程中执行指定的字符串命令，并返回该命令的进程对象。

参数说明

❶ command：一条指定的系统命令。

❷ 返回一个新的 Process 对象，用于管理子进程。

设计过程

（1）创建 FileUtil 类，编写压缩文件为 RAR 文档的方法 toRarFile()，参数 folderPath 为要压缩的文件夹路径，参数 rarFilePath 为 RAR 压缩文件的路径，关键代码如下：

```
public void toRarFile(String folderPath,String rarFilePath) {
        getFiles(folderPath);                                    //调用方法获取文件夹中所有文件的路径
        try {
                File listFile = File.createTempFile("fileList", ".tmp");  //创建临时文件，用于保存压缩文件列表
                StringBuffer fileList = new StringBuffer();
                 for(String filePath:filePaths){
                        fileList.append(filePath+"\n");          //将所有文件路径添加到 StringBuffer 中
                 }

                 FileOutputStream fs = new FileOutputStream(listFile);//根据临时文件，创建文件输出流
                  fs.write(fileList.toString().getBytes());       //将文件路径以字节形式写入临时文件
                 fs.close();
                 File rarFile = new File(rarFilePath);            //根据 RAR 路径创建 File 对象
                 String command = "rar a " + rarFile.getPath() + " @" + listFile.getPath();
                 Runtime runtime = Runtime.getRuntime();          //获取 Runtime 对象
                 process = runtime.exec(command.toString() + "\n");   //执行压缩命令
                 process.getOutputStream().close();               //关闭进程输出流
        } catch (IOException e) {
                 e.printStackTrace();
        }
}
```

（2）创建 index.jsp 页，在该页中获取用户输入的要压缩的文件夹路径和 RAR 文件路径，然后调用 FileUtil

类的 toRarFile()方法，将文件压缩为 RAR 文档，关键代码如下：

```
<%
    request.setCharacterEncoding("UTF-8");                  //设置请求编码
    String submit = request.getParameter("submit");         //获取按钮的值
    String filePath = request.getParameter("filePath");     //获取要压缩的文件夹路径
    String rarfilePath = request.getParameter("rarFile");   //获取 RAR 文件路径
    FileUtil fileUtil = new FileUtil();
    if(submit!=null){                                       //判断是否提交表单
        if(filePath!=null&&!filePath.equals("")&&
            rarfilePath!=null&&!rarfilePath.equals("")){
            fileUtil.toRarFile(filePath,rarfilePath);       //文件压缩为 RAR 文档
        }
    }
%>
```

秘笈心法

调用 Runtime 对象的 exec()方法可以执行一个外部命令并返回命令的进程，这个进程的输出流可以向命令传递参数和其他数据，但是如果这个输出流不关闭，命令就始终不会完成，进程会继续等待输入。就像本实例如果不关闭这个输出流，那么就不会执行压缩命令，除非程序关闭导致输出流被销毁。

实例 314　解压缩 RAR 压缩包

光盘位置：光盘\MR\12\314

高级

实用指数：★★

实例说明

文件解压缩是最常用的数据操作，目前大多数资料和软件都采用 RAR 格式进行压缩并在网站提供下载，经过压缩的资源体积更小、在网络中的传输速度更快。用户从网站下载该文件之后，需要使用 RAR 软件进行解压缩才能获取自己想要的资源。本实例实现对 RAR 压缩包的解压缩功能，可以针对指定的压缩包文件定制解压的目标文件夹，运行效果如图 12.23 所示。



图 12.23　解压缩 RAR 压缩包

关键技术

本小节所介绍关于操作 RAR 文件的所有实例都是通过调用 Rar.exe 命令来执行的，如果读者的计算机中安装了 WinRar，那么在该软件的安装文件夹中会包含 rar.exe 命令文件，本节所有实例需要这个命令才能实现 RAR 文档的操作。下面介绍使用该命令的方法。

（1）复制 RAR 命令文件到项目文件夹

可以找到 WinRar 软件的安装文件夹，把 rar.exe 文件复制到自己的项目中。例如，在 Eclipse 项目中，把 rar.exe 文件复制到根目录，即与 src 文件夹同级，这样程序代码就可以直接调用该命令；也可以把 rar.exe 文件复制到某个文件夹，然后在程序代码中使用绝对路径来调用该命令，但是那样不利于软件的复制与传播。

（2）设置 path 环境变量

另一种方法是复制 WinRar 软件的安装文件夹路径，然后添加到系统的 path 环境变量中。例如，笔者计算机中的安装路径是“C:\Program Files\WinRAR”。把这个文件夹路径作为值添加到 path 环境变量中，如果其左右都有其他变量值，那么使用英文的“;”分号进行分割。例如：

```
;C:\Program Files\WinRAR;
```

把上面的字符串添加到 path 环境变量中，然后重新运行 Eclipse 以使用新的系统环境变量。

注意：path 环境变量的原有内容不要删除，否则会影响其他软件的使用，这包括对 Java 编译命令的影响。

设计过程

（1）创建 FileUtil 工具类，在该类中编写继承自 Thread 类的内部线程类 DeCompressThread，实现调用文件的解压缩命令，关键代码如下：

```
private final class DeCompressThread extends Thread {
        private final String command;
        private DeCompressThread(String command) {
                this.command = command;
        }
        public void run() {
                try {
                        final Process process = Runtime.getRuntime().exec(command);
                        process.getOutputStream().close();
                } catch (IOException e1) {
                        e1.printStackTrace();
                }
        }
}
```

（2）在 FileUtil 类中，编写文件解压缩的方法 enRarFile()，参数 rarFilePath 指的是要解压缩的 RAR 文件路径，参数 targetFolderPath 指的是文件解压路径，在方法中启动 DeCompressThread 线程实现 RAR 文件的解压缩，关键代码如下：

```
public void enRarFile(String rarFilePath,String targetFolderPath) {
    File rarFile = new File(rarFilePath);
    File targetFolder = new File(targetFolderPath);
    if(!targetFolder.exists())                                          //如果文件夹路径不存在，则创建
            targetFolder.mkdirs();
    String command = "rar x " + rarFile + " " + targetFolder + " /y";   //解压缩命令的字符串
    new DeCompressThread(command).start();                              //创建并启动解压缩线程
}
```

（3）创建 index.jsp 页，在该页中获取用于选择的 RAR 文件路径和输入的解压路径，调用 FileUtil 类的 enRarFile()方法，实现 RAR 文件的解压缩，关键代码如下：

```
<%
        request.setCharacterEncoding("UTF-8");                          //设置请求编码
        String submit = request.getParameter("submit");                 //获取按钮的值
        String filePath = request.getParameter("filePath");             //获取要解压的路径
        String rarfilePath = request.getParameter("rarFile");           //获取 RAR 文件路径
        FileUtil fileUtil = new FileUtil();
        if(submit!=null){                                               //判断是否提交表单
                if(filePath!=null&&!filePath.equals("")&&
                        rarfilePath!=null&&!rarfilePath.equals("")){
                        fileUtil.enRarFile(rarfilePath,filePath);       //调用解压 RAR 文件的方法
                }
        }
%>
```

秘笈心法

解压缩动作根据压缩包的大小、包含的文件数量来决定，如果压缩包内容较多、体积较大，解压缩就会变成一个非常耗时的操作，这样的业务是不允许在 GUI 线程中处理的，所以应该采用新的线程来接收文件的解压缩操作，从而把 GUI 线程解放，避免程序界面死锁。

实例 315	文件分卷压缩 光盘位置：光盘\MR\12\315	高级 实用指数：★★

实例说明

把多个文件压缩成一个压缩文档是方便文件的保存与传输，经过压缩后文件内容不变，但是占用磁盘空间更小，这样有利于 Internet 传输，也可以利用移动设备在多个计算机中进行复制。有时对于太大或者太多的文件

进行压缩后体积仍然很大，这种情况下可以把压缩后的文件进行分卷，也就是说压缩后的 RAR 文档不再是一个文件而是多个指定大小的压缩分卷文件，这样可以分别传输部分分卷文件，然后在目的地把各分卷文件解压缩成原有资源文件。本实例就利用 RAR 实现了压缩文档的分卷功能，程序运行效果如图 12.24 所示。

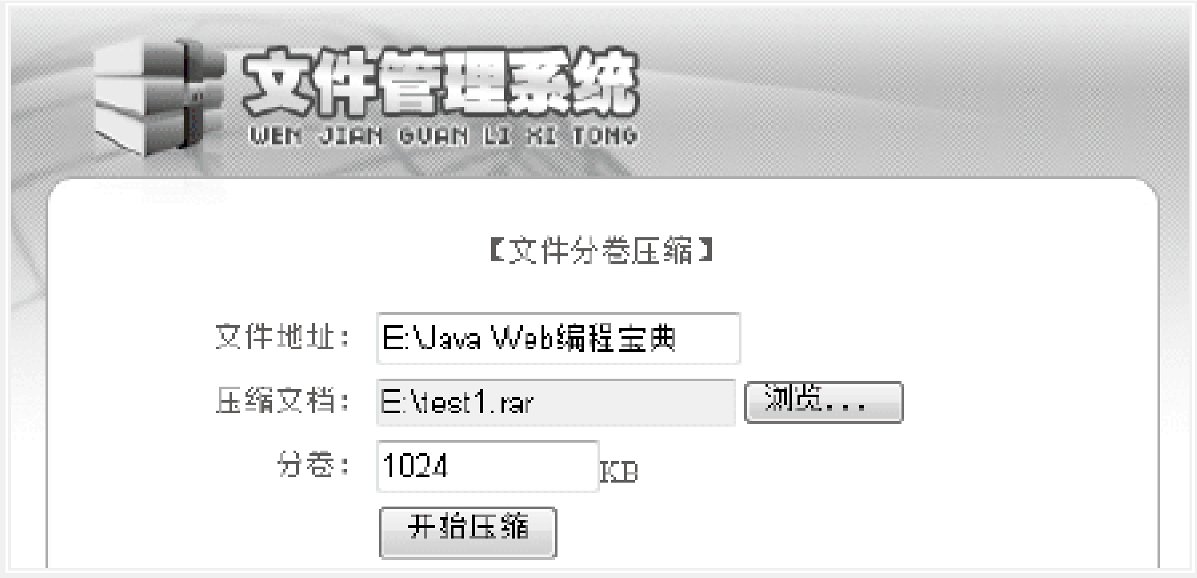


图 12.24　文件分卷压缩

关键技术

本实例和前面讲解的几个用于处理 RAR 文件压缩的实例类似，都是利用 Runtime 类的 exec()方法来执行 WinRAR 软件的 rar 命令。rar 命令的语法格式如下：

```
RAR <命令>[ -<开关> ] <压缩文件>  [ <@列表文件...> ] [ <文件...> ]  [ <解压路径\> ]
```

本实例在实现文件分卷压缩时，主要应用以下命令：

```
rar a -v<大小>[k|b|f|m|M|g|G] rarFilePath @TmpFilePath
```

参数说明

❶ a：添加文件到压缩文件中。

❷ -v：建立分卷，并指定分卷大小。

❸ rarFilePath：压缩的目标 RAR 文件的绝对路径。

❹ TmpFilePath：包含所有文件路径的 Tmp 临时文件的绝对路径。

说明： WinRAR 的控制台命令参数还有很多，如果想进一步了解这些参数的用法，可以查看安装在 WinRAR 软件目录下的 Rar.txt 说明文件。

设计过程

（1）创建工具类 FileUtil，编写执行文件分卷压缩命令的内部线程类 CompressThread，关键代码如下：

```
private final class CompressThread extends Thread {
 private final String command;
     private CompressThread(String command) {
          this.command = command;
     }
        public void run() {
           try {
               Runtime runtime = Runtime.getRuntime();                    //获取 Runtime 对象
               final Process process = runtime.exec(command.toString() + "\n");   //执行压缩命令
               process.getOutputStream().close();                         //关闭进程输出流
           } catch (IOException e) {
               e.printStackTrace();
           }
     }
}
```

（2）编写文件压缩的方法 toRarFile()，参数 folderPath 表示要压缩的文件夹路径，参数 rarFilePath 表示 RAR 压缩文件路径，参数 volumeSize 指分卷压缩文件的大小，以 KB 为单位。在该方法中调用线程类 CompressThread 执行文件的分卷压缩，关键代码如下：

```
public void toRarFile(String folderPath,String rarFilePath,int volumeSize) {
        getFiles(folderPath);                                          //调用获取文件夹中所有文件路径的方法
        try {
              File listFile = File.createTempFile("fileList", ".tmp");       //创建临时文件，用于保存压缩文件列表
              StringBuffer fileList = new StringBuffer();
              for(String filePath:filePaths){                          //遍历保存文件路径的集合
                     fileList.append(filePath+"\n");                   //将所有文件的路径添加到 StringBuffer 中
              }
              FileOutputStream fs = new FileOutputStream(listFile);    //创建文件输出流
              fs.write(fileList.toString().getBytes());                //将文件路径写入临时文件
              fs.close();
              File rarFile = new File(rarFilePath);                    //根据压缩文件的路径创建 File 对象
               String command = "rar a -v"+volumeSize+"k " + rarFile.getPath() + " @"+ listFile.getPath();//创建压缩命令字符串
```

```
            new CompressThread(command).start();                    //创建并启动压缩线程
        } catch (IOException e) {
            e.printStackTrace();
        }
}
```

（3）创建 index.jsp 页，在该页中获取用户输入的请求参数，然后调用 FileUtil 类的 toRarFile()方法，实现文件的分卷压缩，关键代码如下：

```
<%
    request.setCharacterEncoding("UTF-8");                          //设置请求编码
    String submit = request.getParameter("submit");                 //获取按钮的值
    String filePath = request.getParameter("filePath");             //获取要压缩的文件夹路径
    String rarfilePath = request.getParameter("rarFile");           //获取 RAR 文件路径
    String volumeSize = request.getParameter("volumeSize");         //分卷大小
    FileUtil fileUtil = new FileUtil();
    if(submit!=null){                                               //判断是否提交表单
        if(filePath!=null&&!filePath.equals("")&&
            rarfilePath!=null&&!rarfilePath.equals("")&&
            volumeSize!=null&&!volumeSize.equals("")){
            String tmpFilePath = application.getRealPath("/")+"temp";
            File tmpFileFolder = new File(tmpFilePath);
            if(!tmpFileFolder.exists())
                tmpFileFolder.mkdirs();                             //创建临时文件路径
            fileUtil.toRarFile(filePath,rarfilePath,Integer.parseInt(volumeSize));  //调用文件分卷压缩的方法
        }
    }
%>
```

秘笈心法

本实例中的 RAR 命令用到了压缩文件列表参数，主要用于指定添加到压缩包中的文件名称和路径。这些信息如果以命令行的方式添加，会受到命令行长度的限制，所以应该考虑把它们保存到文件中，然后把这个文件作为参数。很明显这个文件需要保存的是临时数据，那么可以考虑通过 File 类创建一个临时文件，这个临时文件在不需要时会被系统删除，从而释放磁盘空间。

实例 316	为 RAR 压缩包添加注释 光盘位置：光盘\MR\12\316	高级 实用指数：★★

实例说明

压缩文件的名称可以简单地说明其用途，但是有些压缩文档的用途类别不好区分，还有一些压缩文档中的文件需要一份使用说明或者文档注释信息。RAR 提供了对压缩文档添加注释的功能，实例运行效果如图 12.25 和图 12.26 所示。



图 12.25　为 RAR 压缩包添加注释



图 12.26　RAR 文件中添加的注释

关键技术

RAR 命令列表中包含一个“c”命令，该命令可以为 RAR 压缩文档添加长度小于 32767 的注释文本。该命

令的说明如下：

命令：c 添加 RAR 压缩文件注释。当压缩文件被处理时注释被显示。文件的注释长度不允许超过 32767 个字节。

例如，为 annotate.rar 文件添加注释可以使用如下命令：

```
rar c annotate.rar
```

执行该命令后，在接下来的输入行中输入注释信息，按 Enter 键结束并添加注释。

另外注释也可以从文件添加，例如：

```
rar c -zinfo.txt annotate.rar
```

该命令会从 info.txt 文件中读取注释信息，然后将注释信息写入到 annotate.rar 文件中。

设计过程

（1）创建 FileUtil 工具类，编写为 RAR 添加注释的方法 addNoteToRarFile()，参数 note 表示要添加的注释信息，参数 rarFile 表示 RAR 文件，关键代码如下：

```
public void addNoteToRarFile(String note,File rarFile) {
    int length = note.getBytes().length;                          //获取注释文本长度
    if (length > 32767) {                                          //限制文本长度
        return;
    }
    try {
         String command = "rar c \"" + rarFile + "\"";
        Process process = Runtime.getRuntime().exec(command);      //执行添加注释命令
        process.getOutputStream().write(note.getBytes());          //把注释文本传递给注释命令
        process.getOutputStream().close();                         //关闭输出流
        process.getInputStream().close();                          //关闭输入流
    } catch (IOException e1) {
        e1.printStackTrace();
    }
}
```

（2）创建 index.jsp 页，在该页中获取用户选择的 RAR 文件的地址和注释信息，然后调用 FileUtil 类的 addNoteToRarFile()方法，实现为 RAR 文档添加注释，关键代码如下：

```
<%
     request.setCharacterEncoding("UTF-8");                         //设置请求编码
     String submit = request.getParameter("submit");                //获取按钮的值
     String rarfilePath = request.getParameter("rarFile");          //获取 RAR 文件路径
     String note = request.getParameter("note");                    //获取注释信息
     FileUtil fileUtil = new FileUtil();
     if(submit!=null){                                              //判断是否提交表单
          if(rarfilePath!=null&&!rarfilePath.equals("")){
               fileUtil.addNoteToRarFile(note,new File(rarfilePath)); //调用添加注释的方法
          }
     }
%>
```

秘笈心法

数据流是对一种资源访问的通道，它会一直占用该资源，所以在程序处理完相应的业务不再需要该资源的数据流时，应该及时关闭，释放资源。

实例 317	获取压缩包详细文件列表 光盘位置：光盘\MR\12\317	高级 实用指数：★★

实例说明

压缩文档常用于保存重要的或者经常传输的资源文件，也有人用压缩文档来保存不常用的资源，以节省磁盘空间。压缩文件可以理解为一个特殊的文件夹，这些特殊的文件夹中保存的是压缩过的文件，而系统的资源

浏览器无法像浏览文件夹那样去浏览 RAR 压缩文档，这给资源的查找带来了不便，本程序实现 RAR 压缩文档中的文件列表解析，在程序界面中就可以看到 RAR 压缩文档中有什么文件，程序运行效果如图 12.27 所示。

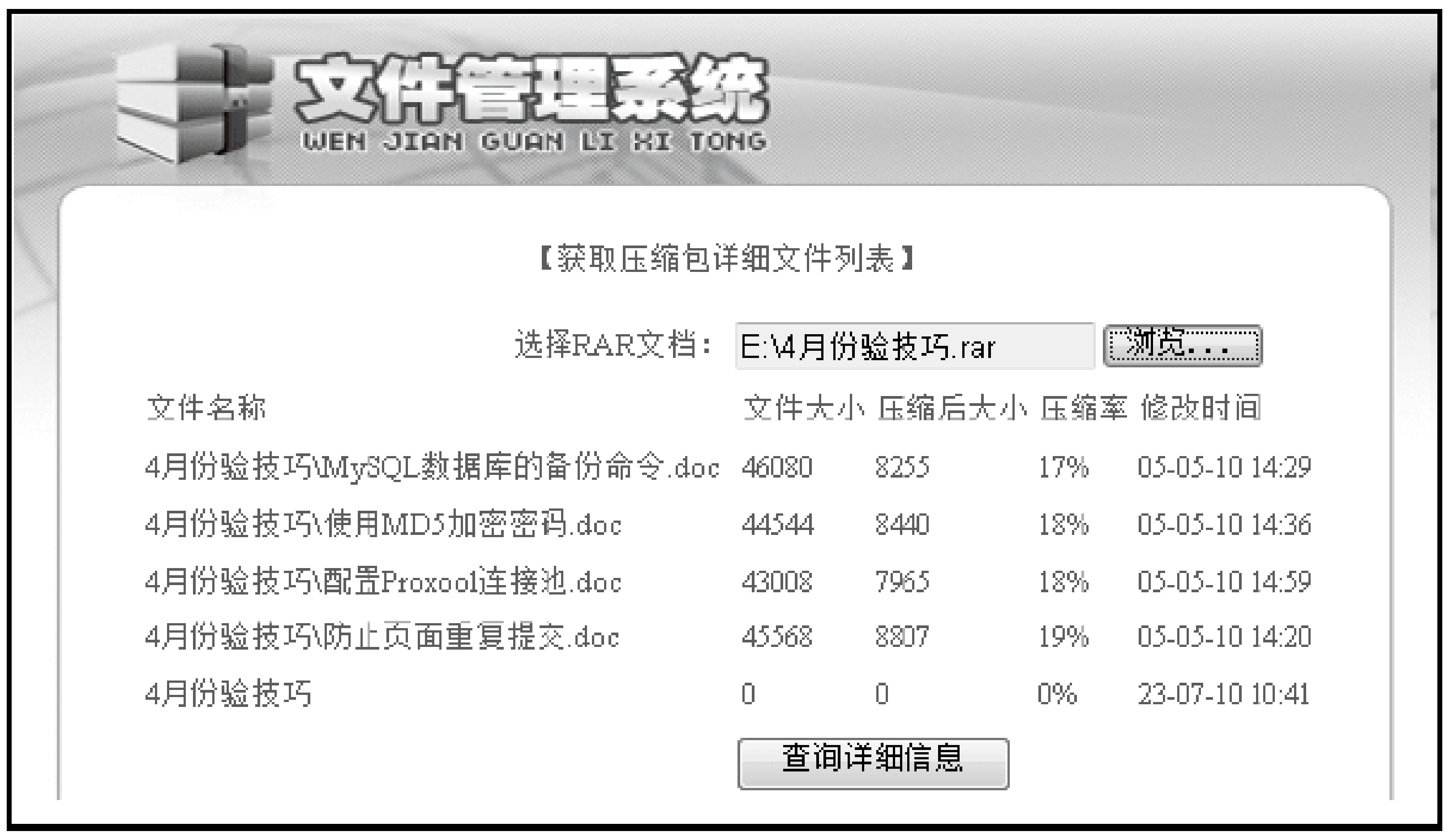


图 12.27　获取压缩包详细文件列表

关键技术

本实例实现 RAR 压缩文件列表的读取与解析，并把详细信息显示在表格控件中，这一切都需要利用 RAR 命令来实现。RAR 有一个命令参数“v”可以显示指定 RAR 文件的列表，而“-c-”开关参数可以不显示 RAR 文档的注释信息，这样方便程序对文件列表的解析。实例中用到的 RAR 完整命令格式如下：

```
rar v -c- "rarFile"
```

参数说明

rarFile：是一个 RAR 压缩文档文件。

设计过程

（1）创建 FileUtil 类，编写获取 RAR 压缩包文件详细列表的方法 getRarDetail()，参数 rarFile 表示 RAR 文件，方法返回封装文件信息列表的 Vector 集合，关键代码如下：

```
public Vector<String> getRarDetail(File rarFile) {
        Vector<String> fileDetails = new Vector<String>();
        try {
            Process process =Runtime. getRuntime() .exec("rar v -c- \"" + rarFile + "\"");//执行文件列表提取命令
            process.getOutputStream().close();     //关闭进程输出流
            Scanner sc = new Scanner(process.getInputStream());
            int count = 0;                         //创建行索引
            while (sc.hasNext()) {
                  String line = sc.nextLine();     //获取文件列表信息的一行
                  //标记起始结束索引
                  if (line.contains("---------------------")) {
                        count = (count == 0 ? count + 1 : -1);
                        continue;
                  }
                  if (count == 0)                  //跳过起始标记
                        continue;
                  if (count == -1)                 //在结束标记终止循环
                        break;
                  if (++count % 2 == 0) {          //获取文件名称
                         fileDetails.add(line);
                  } else {                         //获取文件详细信息
                        fileDetails.add(line);
                  }
            }
            process.getInputStream().close();      //关闭输入流
      } catch (Exception e1) {
            e1.printStackTrace();
      }
      return fileDetails;
}
```

（2）创建 index.jsp 页，在该页中获取用户选择的 RAR 文件，然后调用 FileUtil 类的 getRarDetail()方法，获取文件详细列表，关键代码如下：

```
<%
    Vector<String> details = null;
    request.setCharacterEncoding("UTF-8");              //设置请求编码
    String submit = request.getParameter("submit");      //获取按钮的值
    String rarfilePath = request.getParameter("rarFile");   //获取 RAR 文件路径
    FileUtil fileUtil = new FileUtil();
    if(submit!=null){                                   //判断是否提交表单
        if(rarfilePath!=null&&!rarfilePath.equals("")){
            details = fileUtil.getRarDetail(new File(rarfilePath));
        }
    }
%>
```

（3）在 index.jsp 页中，遍历 Vector 集合，输出文件详细信息到表格，关键代码如下：

```
<%
if(fileDetails!=null){
    for(int i=0;i<fileDetails.size();i++){
 %>
<tr>
    <td><%if(i%2==0){out.println(fileDetails.get(i));}%></td>
    <%
    if(i%2==1){
        String[] info= fileDetails.get(i).split("\\s+");
    %>
    <td><%=info[1] %></td>
    <td><%=info[2] %></td>
    <td><%=info[3] %></td>
    <td><%=info[4] %> <%=info[5] %></td>
    <%} %>
</tr>
<%}} %>
```

秘笈心法

计数器不只是网页上记录用户访问数量的部件，它还可以应用到各种编程环境中。例如，本实例利用计数器对资源的数据进行定位，计数器的作用是区分信息的起始与结束标记。

实例 318	从 RAR 压缩包中删除文件 光盘位置：光盘\MR\12\318	高级 实用指数：★★

实例说明

RAR 压缩文档对于少量文件的压缩与解压缩速度很快，但是如果文件数量和数据太多，使用任何优化技术都无法大幅度提高压缩速度，因为没有数据可以忽略与抛弃。而对于这样大的 RAR 压缩包要删除其中的某个或某些文件，重新解压、整理再压缩是不现实的。为此，本程序通过对 RAR 命令的操作，实现了直接删除 RAR 压缩包中部分文件的功能。程序运行效果如图 12.28 所示，选中复选框选择压缩包文件信息列表中的文件，然后单击“删除文件”按钮，即可将 RAR 压缩包中的文件删除。

关键技术

本实例使用 RAR 的命令实现从 RAR 压缩包中删除指定名称的文件。实例中用到的 RAR 完整命令格式如下：

```
rar  d  -c-  "rarFile"  deleteFile
```

参数说明

❶ rarFile：是一个 RAR 压缩文档文件。

❷ deleteFile：是将要从 RAR 压缩文件中删除的文件。

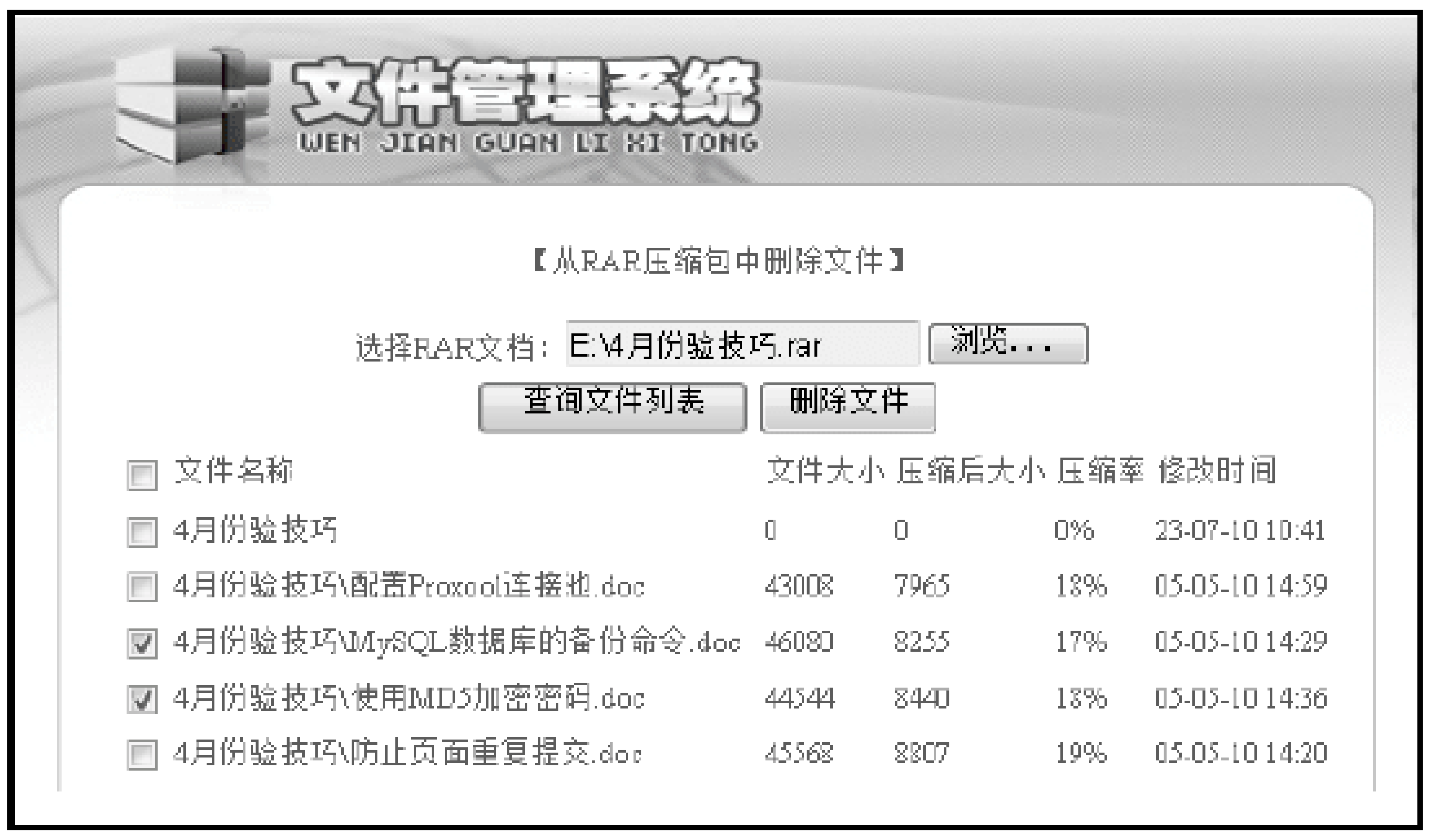


图 12.28　从 RAR 压缩包中删除文件

说明：deleteFile 参数可以使用通配符，如“*”代表所有字符，而“?”代表单个字符。

设计过程

（1）创建 FileUtil 工具类，编写从 RAR 压缩包删除文件的方法 deleteFileFromRar()，参数 rarFile 表示 RAR 文件，参数 filePath 表示要删除的文件路径，关键代码如下：

```
public void deleteFileFromRar(File rarFile,String filePath){
    try {
        Process exec = Runtime.getRuntime().exec("rar d -c- \"" + rarFile + "\" " + filePath);   //执行 RAR 删除命令
        Scanner scan = new Scanner(exec.getInputStream());                                      //创建进程输入流
        while (scan.hasNext()) {                                                                //遍历输入流内容
            scan.nextLine();                                                                    //清空输入流数据
        }
        scan.close();                                                                           //关闭输入流
    } catch (IOException e1) {
        e1.printStackTrace();
    }
}
```

（2）创建 index.jsp 页，在该页中获取用户选中的复选框的值，用户每选中一个复选框，就会调用自定义 JavaScript 函数将复选框的值累加，并以（,）分隔开，该函数的实现比较简单，此处不再详细介绍，具体代码请查看本书附带的光盘。接下来根据用户选中的复选框的值（文件名称）调用 FileUtil 类的 deleteFileFromRar() 方法，删除 RAR 压缩包中的文件，关键代码如下：

```
<%
    Vector<String> fileDetails = null;
    request.setCharacterEncoding("UTF-8");                          //设置请求编码
    String submit = request.getParameter("submit");                 //获取按钮的值
    String delButton = request.getParameter("delButton");
    String rarfilePath = request.getParameter("rarFile");           //获取 RAR 文件路径
    FileUtil fileUtil = new FileUtil();
    if(submit!=null){                                               //单击“查询文件列表”按钮
        if(rarfilePath!=null&&!rarfilePath.equals("")){
            fileDetails = fileUtil.getRarDetail(new File(rarfilePath));
        }
    }
    if(delButton!=null){                                            //单击“删除文件”按钮
        String pathStr = request.getParameter("path");              //获取复选框的值，该值由“,”分隔
        String[] paths = pathStr.split(",");
        for(String path:paths){
            fileUtil.deleteFileFromRar(new File(rarfilePath),path); //调用删除文件的方法
        }
        fileDetails = fileUtil.getRarDetail(new File(rarfilePath)); //重新加载文件列表
    }
%>
```

秘笈心法

在 JDK 1.5 中新增了 Scanner 扫描器类，该类提供了更加灵活的输入流读取方式，并且支持正则表达式，本

身提供的 API 可以读取单个字符、单个类型数据、整行读取等。

实例 319　在压缩文件中查找字符串

光盘位置：光盘\MR\12\319

高级

实用指数：★★

实例说明

计算机中有很多数据文件都使用 RAR 或其他压缩格式进行备份，在需要时来做恢复。作为备份的压缩文件随着资料的基类，压缩内容会越来越多，本实例实现了在压缩文件中搜索字符串的功能，通过这个搜索可以确认某个要查找的资料位于压缩包中的哪个文件中，程序运行效果如图 12.29 所示。

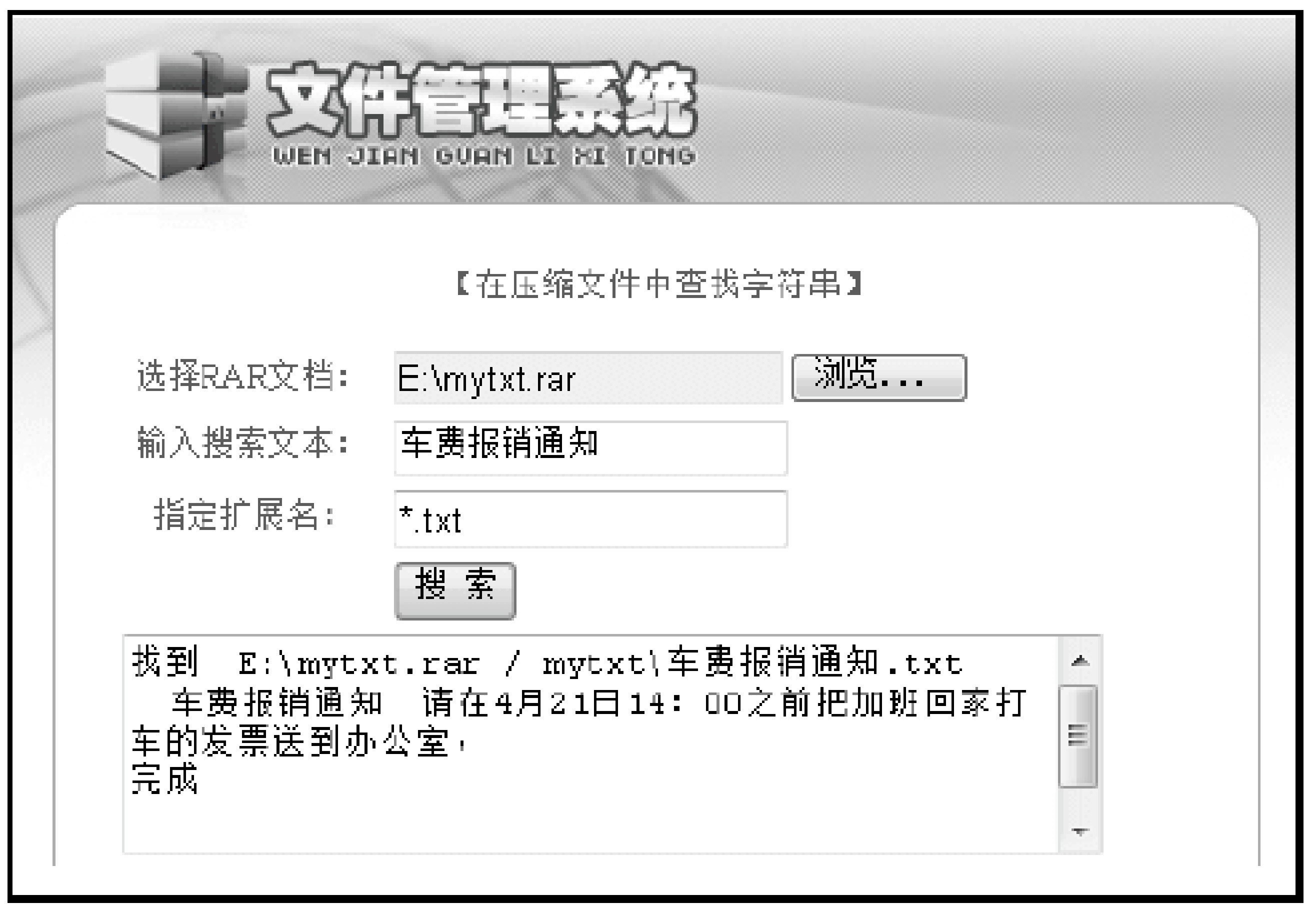


图 12.29　在压缩文件中查找字符串

关键技术

本实例使用 RAR 命令实现从 RAR 压缩包中搜索指定的字符串，并确定该字符串位于某个文件中。实例中用到的 RAR 完整命令格式如下：

```
rar i[i|c]="sText" -c- "rarFile" extName
```

参数说明

❶ sText：是要搜索的文本字符串。

❷ rarFile：是一个 RAR 压缩文档文件。

❸ extName：是要搜索的文件类型，也就是文件的扩展名，可以使用通配符，例如：

```
rar ii="test" -c- "D:\资料.rar" *.txt
```

说明：命令中的 i 是搜索用的，其中还有 i 和 c 两个子命令分别用于指定搜索不区分大小写和区分大小写。

设计过程

（1）创建 FileUtil 工具类，编写在压缩文件中查找字符串的方法 searchRarInfo()，参数 rarFile 表示 RAR 文件，参数 searchStr 表示要查找的字符串，参数 extendStr 表示要查找的文件的扩展名，关键代码如下：

```
public String searchRarInfo(File rarFile,String searchStr,String extendStr){
        StringBuffer str = new StringBuffer();
    int count = 0;
    try {
        Process process = Runtime.getRuntime().exec("rar i" + "i" + "=\"" + searchStr + "\" -c- \"" + rarFile+ "\" " + extendStr); //执行 RAR 命令
        Scanner scan = new Scanner(process.getInputStream());     //获取进程的输入流扫描器
        while (scan.hasNext()) {                                  //遍历进程执行结果
            String line = scan.nextLine();                        //获取单行信息
            if (line.isEmpty())
                count++;
            if (count < 2)                                        //过滤非查询结果
                continue;
            str.append(line + "\n");                              //查询结果添加到 StringBuffer
```

```
            }
        } catch (IOException e1) {
            e1.printStackTrace();
        }
        return str.toString();                                    //返回查询结果字符串
}
```

（2）创建 index.jsp 页，在该页中获取用户选择的 RAR 文件路径、搜索字符串以及文件扩展名，然后调用 FileUtil 类的 searchRarInfo()方法，获取到查询结果字符串，关键代码如下：

```
<%
    String searchInfo = "";                                       //用于保存查询结果字符串
    request.setCharacterEncoding("UTF-8");                        //设置请求编码
    String submit = request.getParameter("submit");               //获取按钮的值
    String rarfilePath = request.getParameter("rarFile");         //获取 RAR 文件路径
    String searchStr = request.getParameter("searchStr");         //获取查询字符串
    String extendStr = request.getParameter("extendStr");         //获取文件扩展名
    FileUtil fileUtil = new FileUtil();
    if(submit!=null){                                             //单击“搜索”按钮
    if(rarfilePath!=null&&!rarfilePath.equals("")&&
            searchStr!=null&&!searchStr.equals("")&&
            extendStr!=null&&!extendStr.equals("")){
            searchInfo = fileUtil.searchRarInfo(new File(rarfilePath),searchStr,extendStr); //调用查询字符串的方法
    }
}
%>
```

秘笈心法

在 JDK 1.5 中对 String 类添加了 isEmpty()方法，该方法可以判断字符串对象的长度是否为 0，以后可以直接调用该方法，而不必先获取字符串长度然后再进行判断。

实例 320	重命名 RAR 压缩包中的文件 光盘位置：光盘\MR\12\320	高级 实用指数：★★

实例说明

RAR 压缩文档可以保存很多资料文件的备份，并且节省磁盘空间，也利于网络传输。但是一个包含上百个资源文件的 RAR 压缩包，经常包含需要修改的内容。例如，某个文件的名称需要修改，如果把所有文件解压缩、改名再压缩会非常繁琐，本程序通过 RAR 的命令实现了压缩包中单个文件的改名操作，程序运行效果如图 12.30 所示。



图 12.30　重命名 RAR 压缩包中的文件

关键技术

本实例使用 RAR 的命令实现重命名 RAR 压缩包中的文件。实例中用到的 RAR 完整命令格式如下：

```
rar rn <rarFile> <sourceFile1> <target1>
```

参数说明

❶ rarFile：是一个 RAR 压缩文档文件。

❷ sourceFile1：是要修改的位于压缩包中的文件名称。

❸ target1：是新的文件名称，源文件将使用这个新名称命名。

说明： 源文件与目标文件是成对出现的，可以出现多个这样的成对组合，那样会同时修改多个源文件的名称为目标文件名称。

设计过程

（1）创建 FileUtil 工具类，编写重命名 RAR 压缩包中文件的方法 renameFileFromRar()，参数 rarFile 指的是 RAR 文件，参数 sourceFileName 指的是压缩包中其中一个文件的名称，参数 newFileName 指的是用户输入的文件新名称，关键代码如下：

```
public void renameFileFromRar(File rarFile,String sourceFileName,String newFileName){
    try {
        int start = sourceFileName.lastIndexOf("\\")+1;                  //查找文件名称中的“\”位置
        //根据文件的父路径和用户输入的新文件名，组合为新的文件名（包括父目录）
        String newName=sourceFileName.substring(0,start)+newFileName;
        Process exec = Runtime.getRuntime().exec("rar rn -c- \"" + rarFile + "\" " + sourceFileName+" "+newName);
        Scanner scan = new Scanner(exec.getInputStream());               //创建进程输入流
        while (scan.hasNext()) {                                          //变量输入流内容
            scan.nextLine();                                              //清空输入流数据
        }
        scan.close();                                                     //关闭输入流
    } catch (IOException e1) {
        e1.printStackTrace();
    }
}
```

（2）创建 index.jsp 页，在该页中获取用户选择的要重命名的文件名和输入的新文件名，然后调用 FileUtil 类的 renameFileFromRar()方法，对 RAR 压缩包中的文件进行重命名，关键代码如下：

```
<%
    Vector<String> fileDetails = null;
    request.setCharacterEncoding("UTF-8");                               //设置请求编码
    String submit = request.getParameter("submit");                      //获取“查询文件列表”按钮的值
    String renameButton = request.getParameter("renameButton");          //获取“重命名”按钮的值
    String rarfilePath = request.getParameter("rarFile");                //获取 RAR 文件路径
    FileUtil fileUtil = new FileUtil();
    if(submit!=null){                                                     //单击“查询文件列表”按钮
        if(rarfilePath!=null&&!rarfilePath.equals("")){
            fileDetails = fileUtil.getRarDetail(new File(rarfilePath));
        }
    }
    if(renameButton!=null){                                               //单击“重命名”按钮
        String pathStr = request.getParameter("path");                   //获取用户选择的文件
        String newName = request.getParameter("newFileName");            //获取新文件名
        fileUtil.renameFileFromRar(new File(rarfilePath),pathStr,newName); //调用重命名文件的方法
        fileDetails = fileUtil.getRarDetail(new File(rarfilePath));      //重新获取 RAR 压缩包的文件列表
    }
%>
```

秘笈心法

本实例在重命名 RAR 压缩包中的文件时，在执行 RAR 重命名的命令时，要重命名的文件名称与修改后的文件名称需要包含文件的路径，否则重命名后的文件在 RAR 压缩包中的路径将会改变。

实例 321

创建自解压 RAR 压缩包

光盘位置：光盘\MR\12\321

高级

实用指数：★★

实例说明

RAR 可以把多个资料文件压缩成一个压缩包文件，这样就可以把大量的资料文件压缩在一起，然后复制到另一个工作室或者其他工作人员那里进行交流，但是如果对方计算机中没有安装相应的软件进行解压缩，那就麻烦了。本实例实现 RAR 文件的改装，将一个 RAR 压缩文件添加自解压模块，这样它就可以自己运行来解压缩数据，而不需要专门的软件，程序运行效果如图 12.31 所示。



图 12.31　创建自解压 RAR 压缩包

关键技术

本实例使用 RAR 的命令实现为指定 RAR 压缩包添加自解压模块的功能，这将使目标 RAR 压缩文件拥有自行解压缩的能力，而不需要其他软件来实现解压缩。实例中用到的 RAR 完整命令格式如下：

```
rar s   <rarFile>
```

参数说明

rarFile：是一个 RAR 压缩文档文件。

例如：

```
rar s –c- -y  资料.rar
```

这个命令是把名称为“资料.RAR”的文件生成可以自动解压缩的“资料.exe”文件。

说明：示例命令中的-c-是不显示 rar 文档注释信息的意思，而-y 是任何问题都回答，本实例会提示是否覆盖已存在的目标文件，那么-y 会默认回答覆盖。

设计过程

（1）创建 FileUtil 工具类，编写创建自解压 RAR 压缩包的方法 createAutoExecFile()，参数 rarFile 表示用户选择的 RAR 压缩包，关键代码如下：

```java
public static String createAutoExecFile(File rarFile) {
    if (rarFile == null)
        return "";
    StringBuffer sb = new StringBuffer();
    try {
        Process process =Runtime. getRuntime().exec("rar s -y -c- " + rarFile);    //为 RAR 文件创建自解压的命令
        Scanner scan=new Scanner(process.getInputStream());
        int count=0;
        while (scan.hasNext()) {                                                   //遍历进程执行结果
            String line = scan.nextLine();                                         //获取单行信息
            if (line.isEmpty())
                count++;
            if (count < 2)                                                         //过滤非查询结果
```

```
                continue;
            sb.append(line + "\n");                                      //查询结果添加到 StringBuffer
        }
    } catch (IOException e1) {
        e1.printStackTrace();
    }
    return sb.toString();                                                //返回查询结果
}
```

（2）创建 index.jsp 页，在该页中获取用户选择的 RAR 压缩包，然后调用 FileUtil 类的 createAutoExecFile()方法，为 RAR 压缩包创建自解压文件，关键代码如下：

```
<%
    String resultInfo = "";                                              //用于保存执行结果的信息
    request.setCharacterEncoding("UTF-8");                               //设置请求编码
    String submit = request.getParameter("submit");                      //获取按钮的值
    String rarfilePath = request.getParameter("rarFile");                //获取 RAR 文件路径
    if(submit!=null){                                                    //单击“创建”按钮
        if(rarfilePath!=null&&!rarfilePath.equals("")){
            resultInfo = FileUtil.createAutoExecFile(new File(rarfilePath));   //调用创建自解压文件的方法
        }
    }
%>
```

秘笈心法

在类中获取资源，通过 Class 类的 getResourceAsStream()方法可以获取指定资源的输入流，该方法是根据当前类的位置在 ClassPath 环境中进行查找的，比指定资源位置要灵活。其方法声明如下：

```
public InputStream getResourceAsStream(String name)
```

在使用程序资源时，建议使用该方法获取图片、音频等资源的输入流，因为它在 ClassPath 变量中查找资源比较灵活。

实例 322　设置 RAR 压缩包密码

光盘位置：光盘\MR\12\322

高级　实用指数：★★

实例说明

RAR 作为目前最流行的压缩文档，经常被用来打包资源在网络中传输，从网络上下载的资源、软件、音频和视频等很多都是经过 RAR 压缩后供用户下载的。但是有些资源需要保密，只有知道密码的人才能够获取压缩包中的文件。RAR 支持对压缩包设置密码的功能，本程序在 RAR 命令的基础上实现了图形化操作的加密程序，程序运行效果如图 12.32 所示。



图 12.32　设置 RAR 压缩包密码

关键技术

本实例使用 RAR 的命令把用户选定的资源文件压缩为 RAR 压缩包并支持密码设置功能，设置密码以后只有通过合法的密码才能解压缩这个 RAR 压缩包。实例中用到的 RAR 完整命令格式如下：

```
rar a –p["password"]    <rarFile>
```

参数说明

❶ password：是要设置的压缩密码。

❷ rarFile：是一个 RAR 压缩文档文件。

例如：

```
rar a -p"mrsoft" -y 资料.rar *.*
```

这个命令是把当前文件夹中所有文件压缩成名称为“资料.rar”的压缩文件，同时设置该压缩文件的密码为“mrsoft”。

说明： 示例命令中使用双引号来包含密码字符串，是为了让输入的密码支持空格字符，这么做，空格字符会把之后的密码当成命令参数。

设计过程

（1）创建 FileUtil 工具类，编写执行 RAR 命令的内部线程类 CompressThread，关键代码如下：

```java
private final class CompressThread extends Thread {
    private String command;
    public CompressThread(String command){
        this.command = command;
    }
    public void run() {
        try {
            Runtime runtime = Runtime.getRuntime();                          //获取 Runtime 对象
            Process process = runtime.exec(command.toString() + "\n");       //执行压缩命令
            process.getOutputStream().close();                               //关闭进程输出流
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

（2）编写文件压缩为 RAR 文档的方法 toRarFile()，参数 folderPath 表示要压缩的文件夹路径，参数 rarFilePath 表示 RAR 文件路径，参数 pwd 表示为压缩文件设置的密码，关键代码如下：

```java
public boolean toRarFile(String folderPath,String rarFilePath,String pwd) {
    getFiles(folderPath);                                                    //调用获取文件夹所有文件路径的方法
    try {
        File listFile = File.createTempFile("fileList", ".tmp");             //创建临时文件，用于保存压缩文件列表
        StringBuffer fileList = new StringBuffer();
        for(String filePath:filePaths){
            fileList.append(filePath+"\n");
        }
        FileOutputStream fs = new FileOutputStream(listFile);                //根据临时文件，创建文件输出流对象
        fs.write(fileList.toString().getBytes());                            //将文件路径写入临时文件
        fs.close();
        File rarFile = new File(rarFilePath);                                //根据 RAR 文件路径创建 File 对象
        String passCommand = "-p\"" + pwd + "\" ";                           //完成密码命令
        String command = "rar a " + passCommand+ rarFile.getPath() + " @" + listFile.getPath();//文件压缩命令，并设置文件密码
        new CompressThread(command).start();                                 //启动线程类，执行文件压缩命令
      return true;
    } catch (IOException e) {
        e.printStackTrace();
        return false;
    }
}
```

（3）创建 index.jsp 页，在该页中获取用户输入的文件夹路径、RAR 压缩包路径以及密码，然后调用 FileUtil 类的 toRarFile()方法，实现文件压缩到 RAR 文档并设置压缩文件的密码，关键代码如下：

```jsp
<%
    request.setCharacterEncoding("UTF-8");                                   //设置请求编码
    String submit = request.getParameter("submit");                          //获取按钮的值
    String rarfilePath = request.getParameter("rarFile");                    //获取 RAR 文件路径
    String filePath = request.getParameter("filePath");                      //获取要压缩的文件夹路径
    String pwd = request.getParameter("pwd");                                //获取用户输入的密码
    FileUtil fileUtil = new FileUtil();
```

```
        if(submit!=null){                                           //判断是否提交表单
                if(filePath!=null&&!filePath.equals("")&&rarfilePath!=null&&!rarfilePath.equals("")){
                        fileUtil.toRarFile(filePath,rarfilePath,pwd);        //调用方法，实现文件压缩并设置压缩包密码
                }
        }
%>
```

秘笈心法

某些程序需要处理的数据量也许会很大，如图片处理，在下一次内存操作之前，可能 JVM 还没有整理出可用内存。但是可以调用 System 类的 gc()方法强制执行垃圾回收机制来获取可用内存。

实例 323　压缩远程文件夹

光盘位置：光盘\MR\12\323

高级

实用指数：★★★☆

实例说明

在前面的范例中，实现了备份网络文件夹的功能。原理是直接将网络文件夹复制到本地文件夹，这样虽然能实现需求，但是效果并不理想。如果能在传输过程中使用压缩技术就好了，这样能提高文件的下载速度。本实例就实现了这样的功能，实例运行效果如图 12.33 所示，用户需要输入想备份的网络文件夹的 URI 地址和备份到哪个文件夹下，单击“开始备份”按钮，会在用户选择的文件夹下生成一个 ZIP 文件。



图 12.33　压缩远程文件夹

关键技术

压缩远程文件夹的实现过程，其实与本地文件夹的压缩基本类似，只不过在获取远程文件夹的文件时，首先需要应用 java.net.URI 类来构建与远程服务器的连接，而且 java.io.File 类的其中一个构造方法也包含一个 URI 类型的构造参数，因此，根据 URI 参数来创建一个 File 对象，就可以应用 File 对象的相应方法获取远程文件夹中的文件。获取远程文件之后，就可以应用相应的文件压缩方法将文件压缩到本地。

设计过程

（1）创建 FileUtil 工具类，编写文件压缩的方法 zipFile()，关键代码如下：

```
public   void zipFile( File targetZipFile, String base) throws IOException {
        FileOutputStream fos = new FileOutputStream(targetZipFile);        //根据给定的 targetZipFile 创建文件输出流对象
        ZipOutputStream zos = new ZipOutputStream(fos);                     //利用文件输出流对象创建 Zip 输出流对象
        byte[] buffer = new byte[1024];
        for (String filePath : filePaths) {                                 //遍历所有要压缩文件的路径
                File currentFile = new File(filePath);
                ZipEntry entry = new ZipEntry(filePath.substring(base.length() + 1, filePath.length()));//利用要压缩文件的相对路径创建 ZipEntry 对象
                FileInputStream fis = new FileInputStream(currentFile);
                zos.putNextEntry(entry);
                int read = 0;
                while ((read = fis.read(buffer)) != -1) {                   //将数据写入到 Zip 输出流中
                        zos.write(buffer, 0, read);
                }
                zos.closeEntry();                                           //关闭 ZipEntry 对象
                fis.close();
        }
```

```
        zos.close();                                                    //释放资源
        fos.close();
}
```

注意：对于需要下载含有中文的文件（夹）的用户，可以考虑使用 Apache 的 ANT 组件实现下载功能，具体参考前面的范例。

（2）编写压缩远程文件夹的方法 backupServerFile()，参数 uri 表示远程文件夹的路径，参数 targetFolder 表示远程文件的压缩路径。在该方法中调用 zipFile()方法，将获取的远程文件夹中的文件进行压缩，关键代码如下：

```
public boolean backupServerFile(String uri ,String targetFolder) {
        try {
                File sourceFile = new File(new URI(uri));                      //根据用户输入的 URI 创建 File 对象
                getFiles(sourceFile.getAbsolutePath());                        //调用获得文件夹中所有文件路径的方法
                File targetFile = new File(targetFolder , sourceFile.getName() + ".zip");   //根据用户选择的文件夹和网络文件夹的名字创建压缩文件
                zipFile(targetFile, sourceFile.getAbsolutePath());             //调用压缩文件的方法，实现压缩功能
                return true;
        } catch (Exception e) {
                e.printStackTrace();
                return false;
        }
}
```

（3）创建 index.jsp 页，在该页中获取用户输入的远程文件夹的地址以及文件压缩的目标路径，调用 FileUtil 类的 backupServerFile()方法，实现远程文件夹的压缩功能，关键代码如下：

```
<%
        request.setCharacterEncoding("UTF-8");                          //设置请求编码
        String submit = request.getParameter("submit");                 //获取按钮的值
        String sourcePath = request.getParameter("sourcePath");         //获取远程文件夹路径
        String targetPath = request.getParameter("targetPath");         //获取目标文件夹路径
        if(submit!=null){                                               //判断是否提交表单
                if(sourcePath!=null&&!sourcePath.equals("")&&targetPath!=null&&!targetPath.equals("")){
                        FileUtil fileUtil = new FileUtil();
                        fileUtil.backupServerFile(sourcePath,targetPath);   //调用压缩远程文件夹的方法
                }
        }
 %>
```

秘笈心法

在文件的传输过程中，使用压缩功能可以提高传输的效率，而且压缩成单个文件也方便管理。压缩可以在下载的过程中进行。

实例 324　压缩存储网页　　高级

光盘位置：光盘\MR\12\324　　实用指数：★★

实例说明

在前面的范例中，实现了下载单个网页的功能，本实例使用压缩技术来下载网页。对于由大量文本组成的网页，压缩技术能大幅度减少下载的流量，提高了网速，程序运行效果如图 12.34 所示。



图 12.34　实例运行效果

说明：用户输入的网址需要包含协议的类型，如 http://，否则会出现异常。

关键技术

在压缩网页内容之前，需要应用 java.net.URL 类的 openConnection()方法与远程网页建立连接，openConnection()方法返回一个 java.net.URLConnection 类型的对象，它代表应用程序和 URL 之间的通信链接。应用 URLConnection 对象来读取和写入此 URL 引用的资源。这些类和方法的具体说明如下：

- public final class URL

类 URL 代表一个统一资源定位符，它是指向互联网“资源”的指针。资源可以是简单的文件或目录，也可以是对更为复杂的对象的引用，如对数据库或搜索引擎的查询。

- openConnection()方法

openConnection()为 URL 类中的方法，通过它可以返回一个 URLConnection 对象，它表示到 URL 所引用的远程对象的连接。

- public abstract class URLConnection

抽象类 URLConnection 是所有类的超类，它代表应用程序和 URL 之间的通信链接。此类的实例可用于读取和写入此 URL 引用的资源。

设计过程

（1）创建 FileUtil 工具类，编写实现下载和压缩网页的方法 zipWebPage()，参数 url 代表用户输入的网址，savePath 代表用户输入的文件夹，关键代码如下：

```
public static void zipWebPage(String url, String savePath) throws IOException {
        URLConnection conn = new URL(url).openConnection();            //利用用户输入的网址创建 URL 链接对象
        InputStream in = conn.getInputStream();                        //从 URLConnection 对象中获得输入流
        FileOutputStream fos = new FileOutputStream(savePath + "download.zip");
        ZipOutputStream zos = new ZipOutputStream(fos);
        byte[] buffer = new byte[1024];
        ZipEntry entry = new ZipEntry("download.html");                //创建名为 download.html 的压缩条目
        zos.putNextEntry(entry);
        int read = 0;
        while ((read = in.read(buffer)) != -1) {                       //写入数据
                zos.write(buffer, 0, read);
        }
        zos.closeEntry();
        in.close();                                                    //释放资源
        zos.close();
        fos.close();
}
```

注意：本程序直接在用户选择的文件夹下创建了一个名为 download.zip 的压缩文件，读者可以根据实际情况进行修改。

（2）创建 index.jsp 页，在该页中获取用户输入的网址和网页保存路径，然后调用 FileUtil 类的 zipWebPage()方法实现网页的下载并压缩保存，关键代码如下：

```
<%
        request.setCharacterEncoding("UTF-8");                         //设置请求编码
        String submit = request.getParameter("submit");                //获取按钮的值
        String downPath = request.getParameter("downPath");            //获取网址
        String savePath = request.getParameter("savePath");            //获取保存路径
        if(submit!=null){                                              //判断是否提交表单
                if(downPath!=null&&!downPath.equals("")&&savePath!=null&&!savePath.equals("")){
                        FileUtil.zipWebPage(downPath,savePath);        //调用压缩存储网页的方法
                }
        }
%>
```

秘笈心法

网页是互联网的重要组成部分，用户在上网的过程中，大部分时间是和网页打交道。如果遇到自己喜欢的网页可以把它保存到本地。此时如果使用压缩技术，就能节约大量的磁盘空间。当需要浏览时，再释放即可。

12.3　文件的批量上传

实例 325	使用 jspSmartUpload 实现文件批量上传 光盘位置：光盘\MR\12\325	高级 实用指数：★★

实例说明

在开发 Web 应用程序或网站时，文件上传的功能是必不可少的，单个文件的上传功能不方便用户的操作，用户每上传一个文件就要提交一次，这样的文件上传功能并不符合用户的实际需求。目前，大多数网站提供的文件上传功能都是批量上传。本实例介绍的是如何应用 jspSmartUpload 实现文件批量上传，程序运行效果如图 12.35 所示。



图 12.35　使用 jspSmartUpload 实现文件批量上传

关键技术

应用 jspSmartUpload 组件实现文件批量上传与单文件上传的思路基本相同。在实现批量上传时，主要应用 jspSmartUpload 类中的 getFiles()方法获取上传的所有文件的 Files 数组，然后通过循环这个文件数组，再通过每个文件表示的 File 对象的 saveAs()方法保存文件即可。

设计过程

（1）创建 index.jsp 页，在该页面中添加文件域表单项，并添加“添加”“删除”“上传”“取消”按钮。用户单击“添加”按钮将触发 onClick 事件，调用自定义的 addMoreRow()函数，在页面中添加新的一行；单击“删除”按钮将触发 onClick 事件，调用自定义的 deleteMoreRow()函数，在页面中删除一行。这两个自定义函数的关键代码如下：

```
<script type="text/javascript">
	function addMoreRow(){									//添加新行
		var perent = event.srcElement.parentNode.parentNode;		//获取触发事件的上两级元素
		var oTable = perent.parentNode.parentNode;				//获取表格元素
		oNewRow = oTable.insertRow();							//在表格中添加一行
		for(i=0;i<perent.cells.length;i++)
		{													//循环添加触发事件上级元素拥有的子元素
			oNewRow.insertCell().innerHTML=perent.cells[i].innerHTML;
		}
		perent.all("DelBtn").disabled=false;					//将父级元素中的“删除”按钮设置为可选状态
```

```
            oNewRow.all("DelBtn").disabled=false;                          //将新添加的行中的“删除”按钮设置为可选状态
        }
        function deleteMoreRow(){                                          //删除行
            var perent=event.srcElement.parentNode.parentNode;             //获取触发事件的上两级元素
            var table=perent.parentNode.parentNode;                        //获取表格元素
            if(table.rows.length>1)                                        //如果当前表格的指定行数大于 1
            {
                table.deleteRow(perent.rowIndex);                          //删除当前行
                if(table.rows.length==1)                                   //如果当前表格行数为 1
                {
                    table.all("DelBtn").disabled=true;                     //将“删除”按钮设置为不可选
                }
            }
        }
</script>
```

（2）创建 HttpServer 的实现类 FileUpload，在该类的 doPost()方法中，应用 jspSmartUpload 组件获取表单请求中的文件对象，实现文件的批量上传。doPost()方法的关键代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        SmartUpload smartUpload= new SmartUpload();                        //创建上传文件的核心对象
        String res = "没有确定要上传的文件！ ";                              //设置返回信息字符串
        String uploadPath = this.getServletContext().getRealPath("/")+"upload";  //在 Web 服务器目录下定义一个上传文件夹
        java.io.File uploadFolder = new java.io.File(uploadPath);          //根据目录创建 File 对象
        if(!uploadFolder.exists())                                         //判断此目录是否存在
            uploadFolder.mkdirs();                                         //如果不存在，则创建
        long maxsize = 1024 * 1024 * 5;                                    //定义上传文件的大小
        try {
            smartUpload.initialize(this.getServletConfig(), request, response);  //初始化 SmartUpload 对象
            smartUpload.upload();                                          //调用 upload()方法
            Files uploadFiles = smartUpload.getFiles();                    //获取所有上传文件
            if (uploadFiles.getSize() > maxsize) {                         //判断上传文件的长度是否大于设置的最大文件长度
                res = "文件大小超出范围！ ";
            }
            for (int i = 0; i < uploadFiles.getCount(); i++) {             //获取遍历上传的所有文件
                File file = uploadFiles.getFile(i);                        //获取每个上传的文件对象
                if ((!file.isMissing()) && (uploadFiles.getSize() < maxsize)) { //如果用户选择了上传文件，并且上传的文件小于限制的大小

                    String fileName = uploadFolder.getName()+"/" +file.getFileName();//定义上传文件的名称的相对路径
                    file.saveAs(fileName, File.SAVEAS_VIRTUAL);            //保存文件，注意参数 fileName 为只能是相对路径下的名称
                    res = "文件上传成功！ ";
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        request.setAttribute("result", res);
        request.getRequestDispatcher("index.jsp").forward(request, response);  //将页面请求转发到 index.jsp
}
```

秘笈心法

在实现上传文件的功能时，需要限制上传文件的大小，否则，如果有些恶意用户上传的文件非常大，会给服务器造成相当大的压力，服务器可能由于读写大量数据导致系统崩溃。

实例 326　使用 commons-fileUpload 实现文件批量上传　高级

光盘位置：光盘\MR\12\326　　实用指数：★★

实例说明

commons-fileUpload 组件是目前应用最广泛的上传组件之一，应用该组件可以很方便地实现文件的批量上

传。在实际应用时，只需要调用该组件中相应的方法即可实现文件的批量上传。本实例将介绍如何使用 commons-fileUpload 组件实现文件的批量上传。程序运行的效果如图 12.36 所示。



图 12.36　使用 commons-fileUpload 实现文件批量上传

关键技术

ServletFileUpload 类是 commons-fileUpload 组件处理文件上传的核心类。ServletFileUpload 类的常用方法如表 12.8 所示。

表 12.8　ServletFileUpload 类的常用方法

方　法　名	作　　用
public boolean isMultipartContent(HttpServletRequest request)	返回 boolean 值 true 或 false，用于判断请求是否为上传文件的请求，主要是判断 form 表单提交请求类型是否为“multipart/form-data”
public List parseRequest(HttpServletRequest request)	该方法从请求中获取上传文件域的 List 集合
public FileItemIterator getItemIterator(HttpServletRequest request)	该方法从请求中获取文件的迭代器

设计过程

（1）创建 index.jsp 页，在该页面中添加文件域表单项，并添加“添加”“删除”“上传”“取消”按钮。用户单击“添加”按钮将触发 onClick 事件，调用自定义的 addMoreRow()函数，在页面中添加新的一行，单击“删除”按钮将触发 onClick 事件，调用自定义的 deleteMoreRow()函数，在页面中删除一行。这两个自定义函数的具体代码请参见实例 325 中的设计过程。

（2）创建 HttpServer 的实现类 FileUploadServlet，在该类的 doPost()方法中，应用 commons-fileUpload 组件获取表单请求中的文件对象，实现文件的批量上传。doPost()方法的关键代码如下：

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
      String uploadPath = this.getServletContext().getRealPath("/")+"upload";   //定义上传文件的服务器路径
      File uploadFolder = new File(uploadPath);                                  //根据该路径创建 File 对象
      if(!uploadFolder.exists())                                                 //如果路径不存在，则创建
            uploadFolder.mkdirs();
      String message = "文件上传成功！";
      try{
            if(ServletFileUpload.isMultipartContent(request)){
                  DiskFileItemFactory factory = new DiskFileItemFactory();       //创建磁盘工厂，用来配置上传组件 ServletFileUpload
                  factory.setSizeThreshold(20*1024);                             //设置内存中允许存储的字节数
                  factory.setRepository(factory.getRepository());                //设置存放临时文件的目录
                  ServletFileUpload upload = new ServletFileUpload(factory);     //创建新的上传文件句柄
                  int maxSize = 5*1024*1024;                                     //定义上传文件的大小
                  List<FileItem> files = upload.parseRequest(request);           //从请求中得到所有上传域列表
                  for(FileItem fileItem:files){                                  //遍历上传文件集合
                        if(!fileItem.isFormField())  {                           //忽略其他不是文件域的所有表单信息
```

```
                        String name = fileItem.getName();
                         if(fileItem.getSize()>maxSize){              //限制文件大小
                            message = "上传文件不得超过 5MB！ ";
                            break;
                        }
                        if((name == null) ||(name.equals(""))&&(fileItem.getSize()==0))
                            continue;
                        File file = new File(uploadPath,name);        //在上传路径中创建文件对象
                        fileItem.write(file);                         //向文件写数据
                    }
                }
            }
        }
        catch(Exception ex){
            ex.printStackTrace();
        }
        request.setAttribute("result", message);                      //将提示信息保存在 request 对象中
        request.getRequestDispatcher("index.jsp").forward(request, response);
}
```

秘笈心法

实际上应用 commons-fileUpload 组件实现的文件批量上传与单文件上传的实现过程是一样的，在后台获得的都是一个 List 集合。需要注意的是，集合中有些元素不一定是文件，它可能是表单提交过来的其他数据，这就需要应用 FileItem 对象的 isFormField()方法判断每个集合元素是否为表单域，这样保证了只有是文件的情况下才执行上传。

第 3 篇

图像与多媒体篇

⏭ 第 13 章　图像生成

⏭ 第 14 章　图像操作

⏭ 第 15 章　多媒体应用

第13章 图像生成

- 绘制图形和文本
- 绘制图案
- 图形的合并运算
- 文字特效
- 图片特效
- 简单的验证码应用
- 复杂的验证码应用
- 生成条形码

13.1　绘制图形和文本

实例 327	绘制直线 光盘位置：光盘\MR\13\327	初级 实用指数：★★★☆

实例说明

在几何中，直线是向两端无限延伸的，本实例所说的绘制直线，实际上是指直线上两点之间的线段，线段在实际生产和生活中经常使用。运行程序，将在网页中显示出绘制的线段，效果如图 13.1 所示。

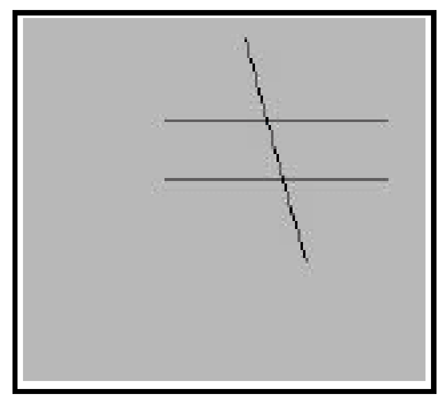


图 13.1　绘制直线

关键技术

本实例主要是通过使用 java.awt.Graphics 类的 drawLine()方法来实现的，创建一个 Graphics 类的实例，表示的是图形上下文对象，用它来绘制基本的形状和文本。drawLine()方法的定义如下：

```
public abstract void drawLine(int x1, int y1, int x2, int y2)
```

参数说明

❶ x1：第一个点的 x 坐标。

❷ y1：第一个点的 y 坐标。

❸ x2：第二个点的 x 坐标。

❹ y2：第二个点的 y 坐标。

设计过程

（1）创建用于生成图像的 Servlet 类 DrawLineServlet，然后在其 service()方法中实现绘制直线，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
	//禁止页面缓存
	response.setHeader("Pragma", "No-cache");
	response.setHeader("Cache-Control", "No-cache");
	response.setDateHeader("Expires", 0);
	response.setContentType("image/jpeg");			//响应格式为 jpeg 图片
	//创建一个指定长宽的图像
	int width=200, height=180;
	BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
	Graphics g = image.getGraphics();			//获取用于处理图形上下文的对象
	g.setColor(new Color(178,186,174));			//设置 RGB 颜色
	g.fillRect(0, 0, width, height);			//填充指定的矩形
	g.setColor(Color.BLACK);				//设置直线的颜色
	g.drawLine(70, 50, 180, 50);				//绘制第一条水平线
	g.drawLine(70, 80, 180, 80);				//绘制第二条水平线
	g.drawLine(110, 10, 140, 120);				//绘制斜线
	g.dispose();						//释放此图形的上下文以及它使用的所有系统资源
	ImageIO.write(image, "JPEG", response.getOutputStream());	//输出 JPEG 格式图片到页面
	response.getOutputStream().flush();			//刷新输出流
	response.getOutputStream().close();			//关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页。在该页中添加一个<img>标签，调用 DrawLineServlet 类生成图像并显示在页面中，关键代码如下：

```
<img alt="" src="DrawLineServlet">
```

秘笈心法

在绘制直线时，如果两个端点的距离很小，就相当于画了一个点，根据这个特点可以在鼠标移动的路径上连续画点，完成各种图形的绘制，从而实现画图板的功能。

实例 328　绘制矩形

光盘位置：光盘\MR\13\328

初级　实用指数：★★★☆

实例说明

矩形在实际生产和生活中经常使用，如书桌的桌面、房屋的门窗等。本实例将通过绘制矩形让读者初步了解 Java 绘图技术。运行程序，将在网页上显示绘制的矩形，效果如图 13.2 所示。

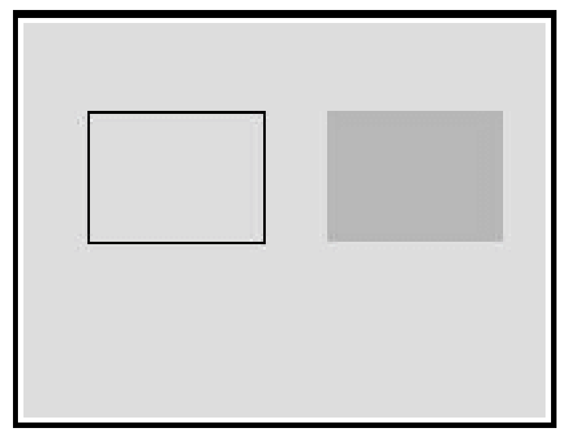


图 13.2　绘制矩形

关键技术

本实例主要是通过使用 Graphics 类的 drawRect()方法和 fillRect()方法来实现的。

（1）使用 Graphics 类的 drawRect()方法绘制的矩形，只有线条而没有填充色，该方法的定义如下：

```
public abstract void drawRect(int x, int y, int width, int height)
```

参数说明

❶ x：矩形左上角的 x 坐标。

❷ y：矩形左上角的 y 坐标。

❸ width：矩形的宽度。

❹ height：矩形的高度。

（2）使用 Graphics 类的 fillRect()方法绘制带填充色矩形，该方法的定义如下：

```
public abstract void fillRect(int x, int y, int width, int height)
```

参数说明

❶ x：填充矩形左上角的 x 坐标。

❷ y：填充矩形左上角的 y 坐标。

❸ width：填充矩形的宽度。

❹ height：填充矩形的高度。

设计过程

（1）创建用于生成矩形图像的 Servlet 类 DrawRectServlet，然后在其 service()方法中实现绘制矩形，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //禁止页面缓存
        response.setHeader("Pragma", "No-cache");
        response.setHeader("Cache-Control", "No-cache");
        response.setDateHeader("Expires", 0);
        response.setContentType("image/JPEG");          //响应格式为 JPEG 图片
        //创建一个指定长宽的图像
        int width=240, height=180;
        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        Graphics g = image.getGraphics();               //获取用于处理图形上下文的对象
        g.setColor(new Color(221,221,221));             //设置第 1 个矩形的 RGB 颜色
        g.fillRect(0, 0, 240, 180);                     //绘制实心矩形
        g.setColor(Color.BLACK);                        //设置第 2 个矩形的颜色
        g.drawRect(30, 40, 80, 60) ;                    //绘制空心矩形
        g.setColor(new Color(178,186,174));             //设置第 3 个矩形的 RGB 颜色
```

```
        g.fillRect(140, 40, 80, 60);                          //绘制实心矩形
        g.dispose();                                          //释放此图形的上下文以及它使用的所有系统资源
        ImageIO.write(image, "JPEG", response.getOutputStream()); //输出 JPEG 格式图片到页面
        response.getOutputStream().flush();                   //刷新输出流
        response.getOutputStream().close();                   //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页。在该页中添加一个<img>标签，调用 DrawRectServlet 类生成图像并显示在页面中，关键代码如下：

```
<img alt="" src="DrawRectServlet">
```

秘笈心法

在调用 Graphics 类的 drawRect()方法和 fillRect()方法绘制矩形之前，可以调用其 setColor()方法为矩形设置颜色。应用 setColor()方法设置颜色时，可以用 java.awt.Color 类的静态常量作为参数，也可以指定一个 RGB 颜色分量的 Color 对象。

实例 329　绘制正方形

光盘位置：光盘\MR\13\329

初级

实用指数：★★★☆

实例说明

本实例将介绍如何在 Java 中绘制正方形。运行程序，将在网页上显示出绘制的正方形，效果如图 13.3 所示。

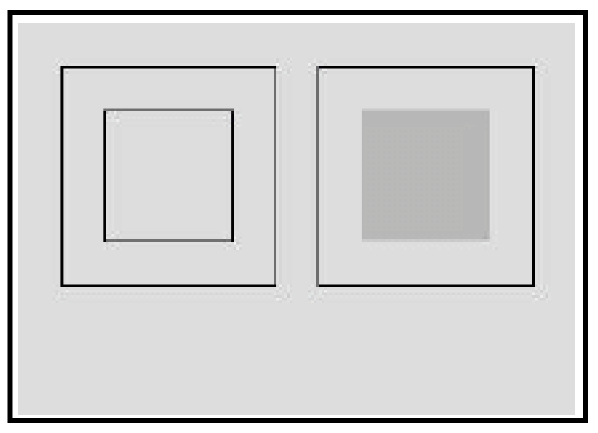


图 13.3　绘制正方形

关键技术

本实例主要是使用 Graphics 类的 drawRect()方法和 fillRect()方法来实现的。

使用 Graphics 类的 drawRect()方法和 fillRect()方法绘制矩形时，如果将这两个方法中表示宽度和高度的参数设置为相同的值，绘制出来的图形就是正方形。

例如：

```
g.drawRect( 30, 20, 120, 120 );                    //在点(30,20)处绘制边长是 120 的正方形
```

设计过程

（1）创建用于生成正方形图像的 Servlet 类 DrawRectServlet，然后在其 service()方法中实现绘制正方形，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response)    throws ServletException, IOException {
        //禁止页面缓存
        response.setHeader("Pragma", "No-cache");
        response.setHeader("Cache-Control", "No-cache");
        response.setDateHeader("Expires", 0);
        response.setContentType("image/JPEG");                //响应格式为 JPEG 图片
        //创建一个指定长宽的图像
        int width=260, height=180;
        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        Graphics g = image.getGraphics();                     //获取用于处理图形上下文的对象
        g.setColor(new Color(221,221,221));                   //设置第 1 个矩形的 RGB 颜色
        g.fillRect(0, 0, 260, 180);                           //绘制实心矩形
        g.setColor(Color.BLACK);                              //设置颜色
        g.drawRect(20, 20, 100, 100);                         //绘制空心正方形
        g.drawRect(40, 40, 60, 60);                           //绘制空心正方形
        g.drawRect(140, 20, 100, 100);                        //绘制空心正方形
        g.setColor(new Color(178,186,174));                   //设置 RGB 颜色
        g.fillRect(160, 40, 60, 60);                          //绘制实心矩形
        g.dispose();                                          //释放此图形的上下文以及它使用的所有系统资源
        ImageIO.write(image, "JPEG", response.getOutputStream()); //输出 JPEG 格式图片到页面
```

```
        response.getOutputStream().flush();                //刷新输出流
        response.getOutputStream().close();                //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页。在该页中添加一个<img>标签，调用 DrawRectServlet 类生成图像并显示在页面中，关键代码如下：

```
<img alt="" src="DrawRectServlet">
```

秘笈心法

在实际项目中绘制柱形图表时，可以使用 fillRect()方法实现。

实例 330	绘制椭圆 光盘位置：光盘\MR\13\330	初级 实用指数：★★★☆

实例说明

本实例将介绍如何在 Java 中绘制椭圆。运行程序，将在网页上显示绘制的椭圆图形，效果如图 13.4 所示。

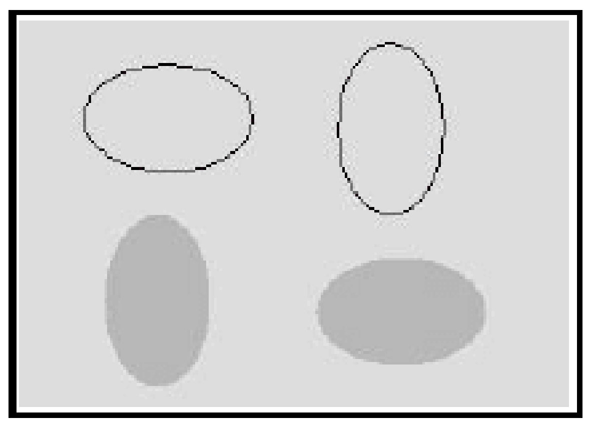


图 13.4　绘制椭圆

关键技术

本实例主要是使用 Graphics 类的 drawOval()方法和 fillOval()方法来实现的。

（1）使用 Graphics 类的 drawOval()方法绘制的椭圆，只有线条而没有填充色，该方法的定义如下：

```
public abstract void drawOval(int x, int y, int width, int height)
```

参数说明

❶ x：要绘制椭圆的左上角的 x 坐标。

❷ y：要绘制椭圆的左上角的 y 坐标。

❸ width：要绘制椭圆的宽度。

❹ height：要绘制椭圆的高度。

（2）使用 Graphics 类的 fillOval()方法绘制带填充色的椭圆，该方法的定义如下：

```
public abstract void fillOval(int x, int y, int width, int height)
```

参数说明

❶ x：要填充椭圆的左上角的 x 坐标。

❷ y：要填充椭圆的左上角的 y 坐标。

❸ width：要填充椭圆的宽度。

❹ height：要填充椭圆的高度。

设计过程

（1）创建用于生成椭圆图形的 Servlet 类 DrawOvalServlet，然后在其 service()方法中实现绘制椭圆图形，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //禁止页面缓存
        response.setHeader("Pragma", "No-cache");
        response.setHeader("Cache-Control", "No-cache");
        response.setDateHeader("Expires", 0);
        response.setContentType("image/JPEG");             //响应格式为 JPEG 图片
        //创建一个指定长宽的图像
        int width=260, height=180;
        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        Graphics g = image.getGraphics();                  //获取用于处理图形上下文的对象
        g.setColor(new Color(221,221,221));                //设置第 1 个矩形的 RGB 颜色
```

```
        g.fillRect(0, 0, 260, 180);                        //绘制实心矩形
        g.setColor(Color.BLACK);                           //设置椭圆的颜色
        g.drawOval(30, 20, 80, 50);                        //绘制空心椭圆
        g.drawOval(150, 10, 50, 80);                       //绘制空心椭圆
        g.setColor(new Color(178,186,174));                //设置椭圆的 RGB 颜色
        g.fillOval(40, 90, 50, 80);                        //绘制实心椭圆
        g.fillOval(140, 110, 80, 50);                      //绘制实心椭圆
        g.dispose();                                       //释放此图形的上下文以及它使用的所有系统资源
        ImageIO.write(image, "JPEG", response.getOutputStream()); //输出 JPEG 格式图片到页面
        response.getOutputStream().flush();                //刷新输出流
        response.getOutputStream().close();                //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页。在该页中添加一个<img>标签，调用 DrawOvalServlet 类生成图像并显示在页面中，关键代码如下：

```
<img alt="" src="DrawOvalServlet">
```

秘笈心法

在绘制椭圆时，如果将 drawOval()方法或 fillOval()方法的后两个参数设置为相同的值，就可以绘制出圆形或填充的圆形。

实例 331　绘制圆弧

光盘位置：光盘\MR\13\331

初级　实用指数：★★★☆

实例说明

本实例演示如何在 Java 中绘制圆弧。运行程序，将在网页上显示出绘制的圆弧，效果如图 13.5 所示。

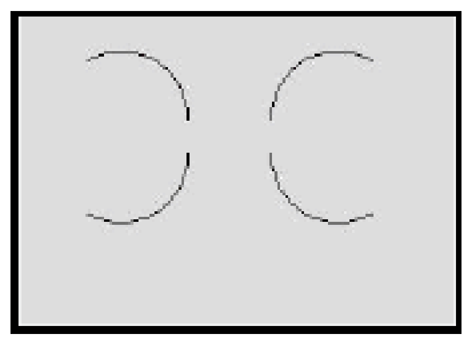


图 13.5　绘制圆弧

关键技术

本实例主要是通过使用 Graphics 类的 drawArc()方法来实现的。使用 Graphics 类的 drawArc()方法绘制圆弧，该方法的定义如下：

```
public abstract void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)
```

参数说明

❶ x：要绘制弧的左上角的 x 坐标。

❷ y：要绘制弧的左上角的 y 坐标。

❸ width：要绘制弧的宽度。

❹ height：要绘制弧的高度。

❺ startAngle：开始角度。

❻ arcAngle：相对于开始角度而言，弧跨越的角度。

设计过程

（1）创建用于生成圆弧图形的 Servlet 类 DrawArcServlet，然后在其 service()方法中实现绘制圆弧图形，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //禁止页面缓存
        response.setHeader("Pragma", "No-cache");
        response.setHeader("Cache-Control", "No-cache");
        response.setDateHeader("Expires", 0);
        response.setContentType("image/JPEG ");            //响应格式为 JPEG 图片
        //创建一个指定长宽的图像
        int width=260, height=180;
        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
```

```
        Graphics g = image.getGraphics();                          //获取用于处理图形上下文的对象
        g.setColor(new Color(221,221,221));                        //设置第 1 个矩形的 RGB 颜色
        g.fillRect(0, 0, 260, 180);                                //绘制实心矩形
        g.setColor(Color.BLACK);                                   //设置颜色
        g.drawArc(20, 20, 80, 80, 0, 120);                         //绘制圆弧
        g.drawArc(20, 40, 80, 80, 0, -120);                        //绘制圆弧
        g.drawArc(150, 20, 80, 80, 180, -120);                     //绘制圆弧
        g.drawArc(150, 40, 80, 80, 180, 120);                      //绘制圆弧
        g.dispose();                                               //释放此图形的上下文以及它使用的所有系统资源
        ImageIO.write(image, "JPEG", response.getOutputStream());  //输出 JPEG 格式图片到页面
        response.getOutputStream().flush();                        //刷新输出流
        response.getOutputStream().close();                        //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页。在该页中添加一个<img>标签，调用 DrawArcServlet 类生成图像并显示在页面中，关键代码如下：

```
<img alt="" src="DrawArcServlet">
```

秘笈心法

在实际开发中，如果需要绘制扇形，可以在使用 drawArc()方法绘制圆弧时，用 drawLine()方法从圆弧的两个端点向圆心画直线，这样就可以画出扇形。

实例 332　绘制指定角度的填充扇形

光盘位置：光盘\MR\13\332

初级

实用指数：★★★☆

实例说明

本实例演示如何在 Java 中绘制指定角度的填充扇形。运行程序，将在网页上显示出绘制的填充扇形，效果如图 13.6 所示。

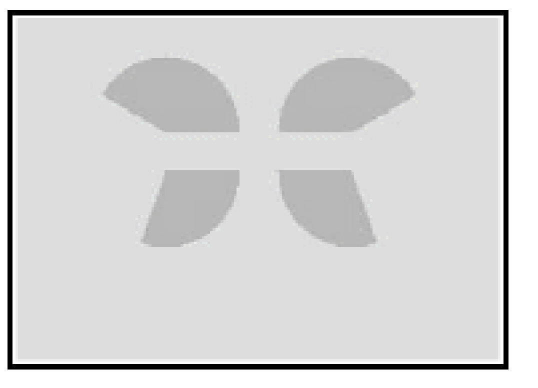


图 13.6　绘制指定角度的填充扇形

关键技术

本实例主要是通过使用 Graphics 类的 fillArc()方法来实现的。使用 Graphics 类的 fillArc()方法绘制填充扇形，该方法的定义如下：

```
public abstract void fillArc(int x, int y, int width, int height, int startAngle, int arcAngle)
```

参数说明

❶ x：要绘制填充扇形的左上角的 x 坐标。

❷ y：要绘制填充扇形的左上角的 y 坐标。

❸ width：要绘制填充扇形的宽度。

❹ height：要绘制填充扇形的高度。

❺ startAngle：开始角度。

❻ arcAngle：相对于开始角度而言，填充扇形的弧跨越的角度。

设计过程

（1）创建用于生成扇形图的 Servlet 类 DrawArcServlet，然后在其 service()方法中实现绘制填充扇形，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //禁止页面缓存
        response.setHeader("Pragma", "No-cache");
        response.setHeader("Cache-Control", "No-cache");
        response.setDateHeader("Expires", 0);
        response.setContentType("image/JPEG ");                    //响应格式为 JPEG 图片
        //创建一个指定长宽的图像
```

```
        int width=260, height=180;
        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        Graphics g = image.getGraphics();                        //获取用于处理图形上下文的对象
        g.setColor(new Color(221,221,221));                      //设置第 1 个矩形的 RGB 颜色
        g.fillRect(0, 0, 260, 180);                              //绘制实心矩形
        g.setColor(new Color(178,186,174));                      //设置扇形的 RGB 颜色
        g.fillArc(40, 20, 80, 80, 0, 150);                       //绘制填充扇形
        g.fillArc(140, 20, 80, 80, 180, -150);                   //绘制填充扇形
        g.fillArc(40, 40, 80, 80, 0, -110);                      //绘制填充扇形
        g.fillArc(140, 40, 80, 80, 180, 110);                    //绘制填充扇形
        g.dispose();                                             //释放此图形的上下文以及它使用的所有系统资源
        ImageIO.write(image, "JPEG", response.getOutputStream()); //输出 JPEG 格式图片到页面
        response.getOutputStream().flush();                      //刷新输出流
        response.getOutputStream().close();                      //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页。在该页中添加一个<img>标签，调用 DrawArcServlet 类生成图像并显示在页面中，关键代码如下：

```
<img alt="" src="DrawArcServlet">
```

秘笈心法

在实际项目中绘制饼形图表时，可以使用 fillArc()方法绘制填充扇形。

实例 333　绘制多边形

光盘位置：光盘\MR\13\333

初级

实用指数：★★★☆

实例说明

本实例演示如何在 Java 中绘制多边形。运行程序，将在网页上显示绘制的多边形，效果如图 13.7 所示。

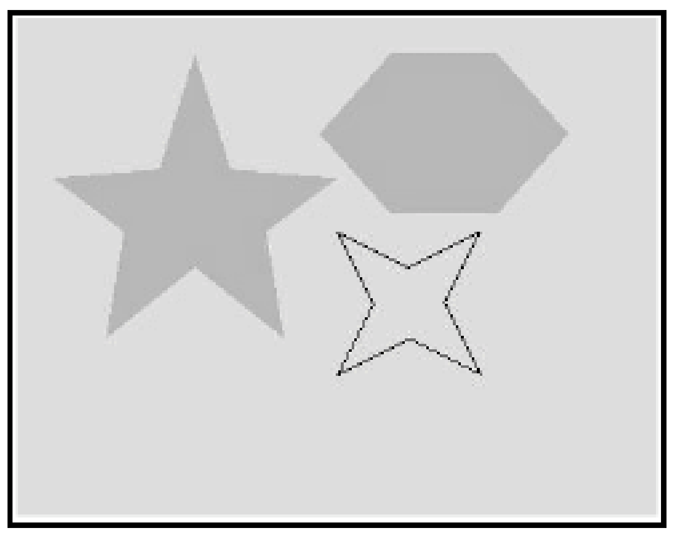


图 13.7　绘制多边形

关键技术

本实例主要是通过使用 Graphics 类的 drawPolygon()方法和 fillPolygon()方法来实现的。

（1）使用 Graphics 类的 drawPolygon()方法绘制的多边形，只有线条而没有填充色，该方法的定义如下：

```
public abstract void drawPolygon(int[] xPoints, int[] yPoints, int nPoints)
```

参数说明

❶ xPoints：要绘制多边形的 x 坐标数组。

❷ yPoints：要绘制多边形的 y 坐标数组。

❸ nPoints：要绘制多边形的顶点总数。

（2）使用 Graphics 类的 fillPolygon()方法绘制带填充色的多边形，该方法的定义如下：

```
public abstract void fillPolygon(int[] xPoints, int[] yPoints, int nPoints)
```

参数说明

❶ xPoints：要绘制填充多边形的 x 坐标数组。

❷ yPoints：要绘制填充多边形的 y 坐标数组。

❸ nPoints：要绘制填充多边形的顶点总数。

设计过程

（1）创建用于生成多边形的 Servlet 类的 DrawPolyServlet，然后在其 service()方法中实现绘制多边形，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //禁止页面缓存
        response.setHeader("Pragma", "No-cache");
        response.setHeader("Cache-Control", "No-cache");
        response.setDateHeader("Expires", 0);
        response.setContentType("image/JPEG");                    //响应格式为 JPEG 图片
        //创建一个指定长宽的图像
        int width=360, height=280;
        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        Graphics g = image.getGraphics();                         //获取用于处理图形上下文的对象
        g.setColor(new Color(221,221,221));                       //设置第 1 个矩形的 RGB 颜色
        g.fillRect(0, 0, 360, 280);                               //绘制实心矩形
        g.setColor(new Color(178,186,174));                       //设置多边形的 RGB 颜色
        int[] x1 = { 100,120,180,140,150,100,50,60,20,80 };       //多边形的横坐标
        int[] y1 = { 20,85,90,120,180,140,180,120,90,85 };        //多边形的纵坐标
        int n1 = 10;                                              //多边形的边数
        g.fillPolygon(x1, y1, n1);                                //绘制多边形
        int[] x2 = { 210, 270, 310, 270, 210, 170 };              //多边形的横坐标
        int[] y2 = { 20, 20, 65, 110, 110, 65 };                  //多边形的纵坐标
        int n2 = 6;                                               //多边形的边数
        g.fillPolygon(x2, y2, n2);                                //绘制实心多边形
        int[] x3 = { 180, 220, 260, 240, 260, 220, 180, 200 };    //多边形的横坐标
        int[] y3 = { 120, 140, 120, 160, 200, 180, 200, 160 };    //多边形的纵坐标
        int n3 = 8;                                               //多边形的边数
        g.setColor(Color.BLACK);
        g.drawPolygon(x3, y3, n3);                                //绘制多边形
        g.dispose();                                              //释放此图形的上下文以及它使用的所有系统资源
        ImageIO.write(image, "JPEG", response.getOutputStream()); //输出 JPEG 格式图片到页面
        response.getOutputStream().flush();                       //刷新输出流
        response.getOutputStream().close();                       //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页。在该页中添加一个<img>标签，调用 DrawPolyServlet 类生成图像并显示在页面中，关键代码如下：

```
<img alt="" src="DrawPolyServlet">
```

秘笈心法

由于绘制多边形需要各顶点 x 坐标和 y 坐标的数组。因此，可以在草纸上把图形画出来，然后再根据图形定义坐标点就容易多了。

实例 334	绘制二次曲线 光盘位置：光盘\MR\13\334	初级 实用指数：★★★☆

实例说明

本实例演示如何在 Java 中绘制二次曲线。运行程序，将在网页上显示绘制的二次曲线，效果如图 13.8 所示。

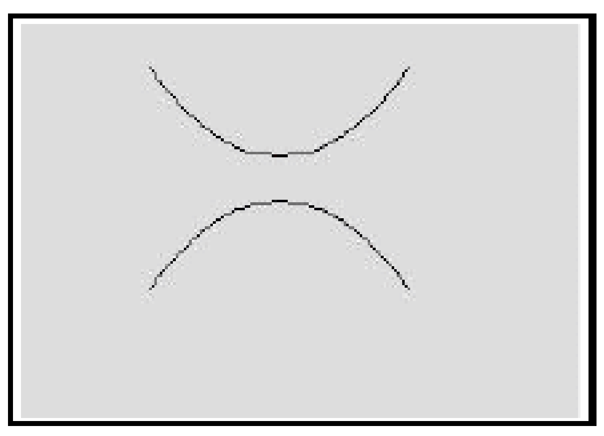


图 13.8　绘制二次曲线

关键技术

本实例主要是通过使用 Graphics2D 类的 draw()方法和使用 QuadCurve2D.Double 类创建二次曲线对象来实现的。

（1）使用 Graphics2D 类的 draw()方法，并将 QuadCurve2D.Double 类创建的二次曲线对象，作为 draw()方法的参数，实现绘制二次曲线的操作，draw()方法的定义如下：

```
public abstract void draw (Shape shape)
```

参数说明

shape：要绘制的形状。

（2）使用 QuadCurve2D.Double 类创建二次曲线对象，其构造方法的定义如下：

```
public QuadCurve2D.Double(double x1, double y1, double ctrlx, double ctrly, double x2, double y2)
```

参数说明

❶ x1：起始点的 x 坐标。

❷ y1：起始点的 y 坐标。

❸ ctrlx：控制点的 x 坐标。

❹ ctrly：控制点的 y 坐标。

❺ x2：结束点的 x 坐标。

❻ y2：结束点的 y 坐标。

设计过程

（1）创建用于生成二次曲线图形的 Servlet 类 DrawQuadCurve2DServlet，然后在其 service()方法中实现绘制二次曲线图形，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //禁止页面缓存
        response.setHeader("Pragma", "No-cache");
        response.setHeader("Cache-Control", "No-cache");
        response.setDateHeader("Expires", 0);
        response.setContentType("image/JPEG");              //响应格式为 JPEG 图片
        //创建一个指定长宽的图像
        int width=260, height=180;
        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        Graphics g = image.getGraphics();                   //获取用于处理图形上下文的对象
        g.setColor(new Color(221,221,221));                 //设置第 1 个矩形的 RGB 颜色
        g.fillRect(0, 0, 260, 180);                         //绘制实心矩形
        Graphics2D g2=(Graphics2D)g;                        //获得 Graphics2D 对象
        g2.setColor(Color.BLACK);
        //创建二次曲线，其中点（120,100）是控制点坐标，点（60,20）是起始点坐标，点（180,20）是终点坐标
        QuadCurve2D.Double quadCurve1 = new QuadCurve2D.Double(60,20,120,100,180,20);
        g2.draw(quadCurve1);                                //绘制二次曲线
        //创建二次曲线，其中点（120,40）是控制点坐标，点（60,120）是起始点坐标，点（180,120）是终点坐标
        QuadCurve2D.Double quadCurve2 = new QuadCurve2D.Double(60,120,120,40,180,120);
        g2.draw(quadCurve2);
        g.dispose();                                        //释放此图形的上下文以及它使用的所有系统资源
        g2.dispose();
        ImageIO.write(image, "JPEG", response.getOutputStream()); //输出 JPEG 格式图片到页面
        response.getOutputStream().flush();                 //刷新输出流
        response.getOutputStream().close();                 //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页，在该页中添加一个<img>标签，调用 DrawQuadCurve2DServlet 类生成图像并显示在页面中，关键代码如下：

```
<img alt="" src="DrawQuadCurve2DServlet">
```

秘笈心法

绘制二次曲线时，可以使用 QuadCurve2D.Double 类和 QuadCurve2D.Float 类，其中 QuadCurve2D.Float 类更节省内存空间。

实例 335	绘制三次曲线 光盘位置：光盘\MR\13\335	初级 实用指数：★★★☆

实例说明

本实例演示如何在 Java 中绘制三次曲线。运行程序，将在网页上显示出绘制的三次曲线，效果如图 13.9 所示。

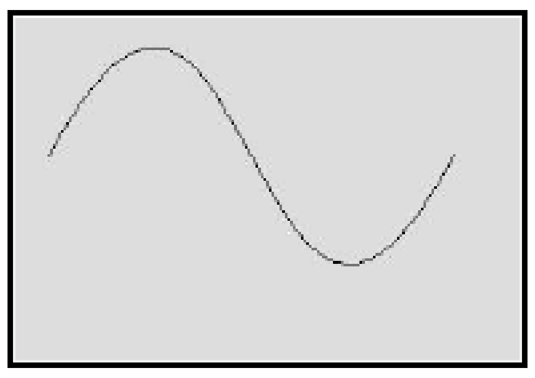


图 13.9　绘制三次曲线

关键技术

本实例主要是通过使用 Graphics2D 类的 draw()方法和使用 CubicCurve2D.Double 类创建三次曲线对象来实现的。

使用 CubicCurve2D.Double 类创建三次曲线对象，其构造方法的定义如下：

```
public CubicCurve2D.Double(double x1, double y1, double ctrlx1, double ctrly1, double ctrlx2, double ctrly2, double x2, double y2)
```

参数说明

❶ x1：起始点的 x 坐标。

❷ y1：起始点的 y 坐标。

❸ ctrlx1：第一个控制点的 x 坐标。

❹ ctrly1：第一个控制点的 y 坐标。

❺ ctrlx2：第二个控制点的 x 坐标。

❻ ctrly2：第二个控制点的 y 坐标。

❼ x2：结束点的 x 坐标。

❽ y2：结束点的 y 坐标。

设计过程

（1）创建用于生成三次曲线图形的 Servlet 类 DrawQuadCurve2DServlet，然后在其 service()方法中实现绘制三次曲线图形，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //禁止页面缓存
        response.setHeader("Pragma", "No-cache");
        response.setHeader("Cache-Control", "No-cache");
        response.setDateHeader("Expires", 0);
        response.setContentType("image/JPEG");                    //响应格式为 JPEG 图片
        //创建一个指定长宽的图像
        int width=300, height=200;
        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        Graphics g = image.getGraphics();                         //获取用于处理图形上下文的对象
        g.setColor(new Color(221,221,221));                       //设置第 1 个矩形的 RGB 颜色
        g.fillRect(0, 0, 300, 200);                               //绘制实心矩形
        Graphics2D g2=(Graphics2D)g;                              //获得 Graphics2D 对象
        g2.setColor(Color.BLACK);
        //创建三次曲线，其中点（140,-140）和点（140,300）是控制点坐标，点（20,80）是起始点坐标，点（260,80）是终点坐标
        CubicCurve2D.Double cubicCurve = new CubicCurve2D.Double(20,80,140,-140,140,300,260,80);
        g2.draw(cubicCurve);                                      //绘制三次曲线
        g.dispose();                                              //释放此图形的上下文以及它使用的所有系统资源
        g2.dispose();
        ImageIO.write(image, "JPEG", response.getOutputStream()); //输出 JPEG 格式图片到页面
        response.getOutputStream().flush();                       //刷新输出流
        response.getOutputStream().close();                       //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页。在该页中添加一个<img>标签，调用 DrawQuadCurve2DServlet 类生成图像并显示在页面中，关键代码如下：

```
<img alt="" src="DrawQuadCurve2DServlet">
```

秘笈心法

绘制三次曲线时，可以使用 CubicCurve2D.Double 类和 CubicCurve2D.Float 类，其中 CubicCurve2D.Float

类更节省内存空间。

实例 336	绘制文本 光盘位置：光盘\MR\13\336	初级 实用指数：★★★☆

实例说明

本实例演示如何在 Java 中绘制文本。运行程序，将在网页上显示绘制的文本，效果如图 13.10 所示。

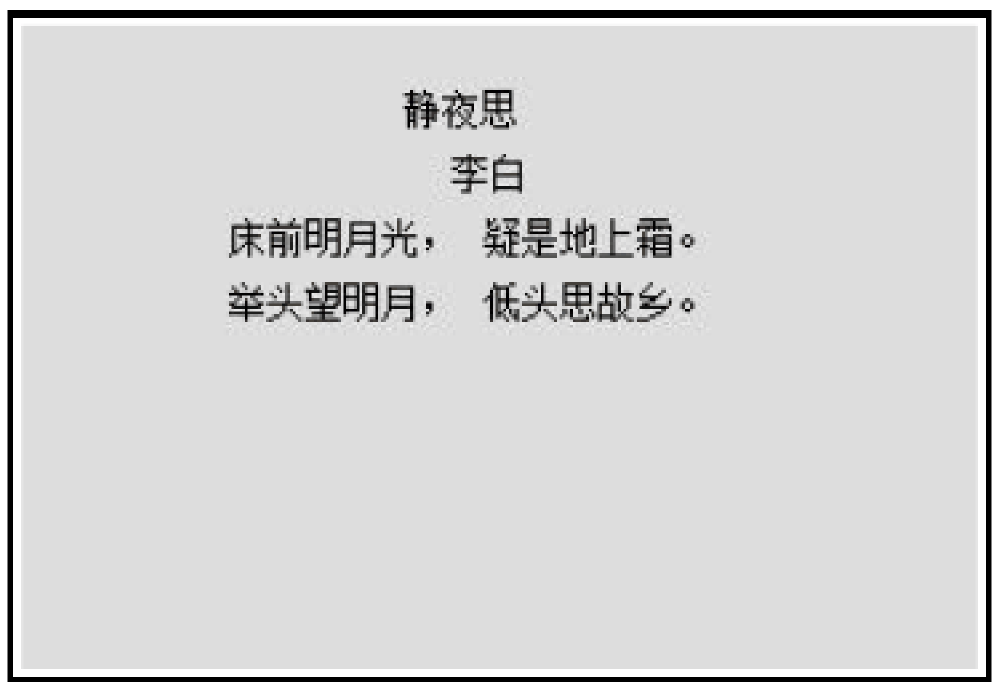


图 13.10　绘制文本

关键技术

本实例主要是使用 Graphics 类的 drawString()方法来实现的。使用 Graphics 类的 drawString()方法绘制文本，该方法的定义如下：

```
public abstract void drawString(String str, int x, int y)
```

参数说明

❶ str：绘制的文本内容。

❷ x：绘制点的 x 坐标。

❸ y：绘制点的 y 坐标。

设计过程

（1）创建用于绘制文本的 Servlet 类 DrawStringServlet，然后在其 service()方法中实现绘制文本，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //禁止页面缓存
        response.setHeader("Pragma", "No-cache");
        response.setHeader("Cache-Control", "No-cache");
        response.setDateHeader("Expires", 0);
        response.setContentType("image/JPEG");                //响应格式为 JPEG 图片
        //创建一个指定长宽的图像
        int width=300, height=200;
        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        Graphics g = image.getGraphics();                     //获取用于处理图形上下文的对象
        g.setColor(new Color(221,221,221));                   //设置第 1 个矩形的 RGB 颜色
        g.fillRect(0, 0, 300, 200);                           //绘制实心矩形
        g.setColor(Color.BLACK);
        String title = "静夜思";
         int x = 120;                                         //文本位置的横坐标
        int y = 30;                                           //文本位置的纵坐标
        g.drawString(title, x, y);                            //绘制文本
        String author = "李白";
        int x1 = 135;
        int y1 = 50;
        g.drawString(author, x1, y1);                         //绘制文本
        ……//此处省略了部分代码
```

```
        g.dispose();                                          //释放此图形的上下文以及它使用的所有系统资源
        ImageIO.write(image, "JPEG", response.getOutputStream()); //输出 JPEG 格式图片到页面
        response.getOutputStream().flush();                   //刷新输出流
        response.getOutputStream().close();                   //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页。在该页中添加一个<img>标签，调用 DrawStringServlet 类生成图像并显示在页面中，关键代码如下：

```
<img alt="" src="DrawStringServlet">
```

秘笈心法

对于一些有可能侵权的图片，或者要作为宣传的图片，可以通过使用 drawString()方法，将用到的文字绘制到图片上，从而得到所谓的“水印文字”。

实例 337 设置文本的字体

光盘位置：光盘\MR\13\337

初级

实用指数：★★★☆

实例说明

本实例演示在 Java 中绘制文本时，如何设置文本的字体，其中包括字体的名称、大小和样式，运行程序，效果如图 13.11 所示。



图 13.11　设置文本字体的效果

关键技术

本实例主要是使用 Graphics 类的 setFont()方法和使用 Font 类创建字体对象来实现的。

（1）使用 Graphics 类的 setFont()方法，并将 Font 类创建的字体对象作为 setFont()方法的参数，实现为文本设置字体的操作，setFont()方法的定义如下：

```
public abstract void setFont(Font font)
```

参数说明

font：为文本设置的字体对象。

（2）使用 Font 类创建字体对象，其构造方法的定义如下：

```
public Font(String name, int style, int size)
```

参数说明

❶ name：字体的名称。

❷ style：字体的样式。

❸ size：字体的大小。

设计过程

（1）创建用于设置文本字体的 Servlet 类 DrawStringServlet，然后在其 service()方法中实现设置文本的字体，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //禁止页面缓存
        response.setHeader("Pragma", "No-cache");
        response.setHeader("Cache-Control", "No-cache");
        response.setDateHeader("Expires", 0);
        response.setContentType("image/JPEG");                //响应格式为 JPEG 图片
        //创建一个指定长宽的图像
        int width=330, height=200;
        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        Graphics g = image.getGraphics();                     //获取用于处理图形上下文的对象
        g.setColor(new Color(221,221,221));                   //设置第 1 个矩形的 RGB 颜色
```

```
        g.fillRect(0, 0, 330, 200);                                    //绘制实心矩形
        g.setColor(Color.BLACK);
        String value = "明日编程词典社区";
        int x = 40;                                                    //文本位置的横坐标
        int y = 50;                                                    //文本位置的纵坐标
        Font font = new Font("华文行楷", Font.BOLD + Font.ITALIC, 26);  //创建字体对象
        g.setFont(font);                                               //设置字体
        g.drawString(value, x, y);                                     //绘制文本
        value = "http://community.mrbccd.com";
        x = 10;                                                        //文本位置的横坐标
        y = 100;                                                       //文本位置的纵坐标
        font = new Font("宋体", Font.BOLD, 20);                         //创建字体对象
        g.setFont(font);                                               //设置字体
        g.drawString(value, x, y);                                     //绘制文本
        g.dispose();                                                   //释放此图形的上下文以及它使用的所有系统资源
        ImageIO.write(image, "JPEG", response.getOutputStream());      //输出 JPEG 格式图片到页面
        response.getOutputStream().flush();                            //刷新输出流
        response.getOutputStream().close();                            //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页，在该页中添加一个<img>标签，调用 DrawStringServlet 类生成图像并显示在页面中，关键代码如下：

```
<img alt="" src="DrawStringServlet">
```

秘笈心法

在绘制文本的同时，经常需要设置文本的字体样式，以达到醒目的效果，字体样式包括粗体样式 Font.BOLD、斜体样式 Font.ITALIC 和普通样式 Font.PLAIN，这些字体样式可以单独设置，也可以组合使用，在组合使用时需要用连接符“+”进行连接，如粗斜体样式为“Font.BOLD+Font.ITALIC”。

实例 338　设置文本和图形的颜色

初级

光盘位置：光盘\MR\13\338

实用指数：★★★☆

实例说明

本实例演示在 Java 中绘制文本和图形时，如何设置文本和图形的颜色。运行程序，效果如图 13.12 所示。

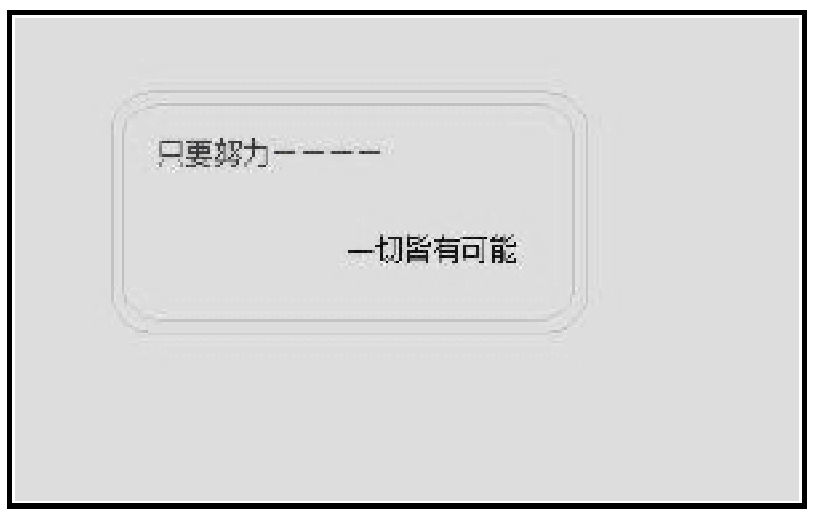


图 13.12　设置文本和图形颜色的效果

关键技术

本实例主要是通过使用 Graphics 类的 setColor()方法和使用 Color 类创建颜色对象来实现的。

（1）使用 Graphics 类的 setColor()方法，并将 Color 类创建的颜色对象作为 setColor()方法的参数，实现为文本和图形设置颜色的操作，setColor()方法的定义如下：

```
public abstract void setColor(Color color)
```

参数说明

color：为文本或图形设置的颜色对象。

（2）使用 Color 类创建颜色对象，其构造方法的定义如下：

```
public Color(int r, int g, int b)
```

参数说明

❶ r：RGB 颜色的 R 值。

❷ g：RGB 颜色的 G 值。

❸ b：RGB 颜色的 B 值。

提示：Color 类提供了多个重载的构造方法，用户可以根据需要进行选择。

设计过程

（1）创建用于设置文本和图形颜色的 Servlet 类 DrawImageServlet，然后在其 service()方法中实现设置文本和图形的颜色，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //禁止页面缓存
        response.setHeader("Pragma", "No-cache");
        response.setHeader("Cache-Control", "No-cache");
        response.setDateHeader("Expires", 0);
        response.setContentType("image/JPEG");                    //响应格式为 JPEG 图片
        //创建一个指定长宽的图像
        int width=330, height=200;
        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        Graphics g = image.getGraphics();                         //获取用于处理图形上下文的对象
        g.setColor(new Color(221,221,221));                       //设置第 1 个矩形的 RGB 颜色
        g.fillRect(0, 0, 330, 200);                               //绘制实心矩形
        String value = "只要努力————";
        int x = 60;                                               //文本位置的横坐标
        int y = 60;                                               //文本位置的纵坐标
        Color color = new Color(255,0,0);                         //创建颜色对象
        g.setColor(color);                                        //设置颜色
        g.drawString(value, x, y);                                //绘制文本
        value = "一切皆有可能";
        x = 140;                                                  //文本位置的横坐标
        y = 100;                                                  //文本位置的纵坐标
        color = new Color(0,0,255);                               //创建颜色对象
        g.setColor(color);                                        //设置颜色
        g.drawString(value, x, y);                                //绘制文本
        color = Color.ORANGE;                                     //通过 Color 类的字段获得颜色对象
        g.setColor(color);                                        //设置颜色
        g.drawRoundRect(40,30,200,100,40,30);                     //绘制圆角矩形
        g.drawRoundRect(45,35,190,90,36,26);                      //绘制圆角矩形
        g.dispose();                                              //释放此图形的上下文以及它使用的所有系统资源
        ImageIO.write(image, "JPEG", response.getOutputStream()); //输出 JPEG 格式图片到页面
        response.getOutputStream().flush();                       //刷新输出流
        response.getOutputStream().close();                       //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页。在该页中添加一个<img>标签，调用 DrawImageServlet 类生成图像并显示在页面中，关键代码如下：

```
<img alt="" src="DrawImageServlet">
```

秘笈心法

在绘制文本和图形时，除了使用 Color 类的构造方法创建颜色对象处，还可以使用 Color 类提供的字段获得颜色对象，如红色为 Color.RED 或 Color.red。

13.2 绘 制 图 案

实例 339	绘制五环图案 光盘位置：光盘\MR\13\339	初级 实用指数：★★★☆

实例说明

本实例演示大家所熟悉的奥林匹克运动会的会徽，即五环图案的绘制。运行程序，将在网页上显示绘制的

五环图案，效果如图 13.13 所示。

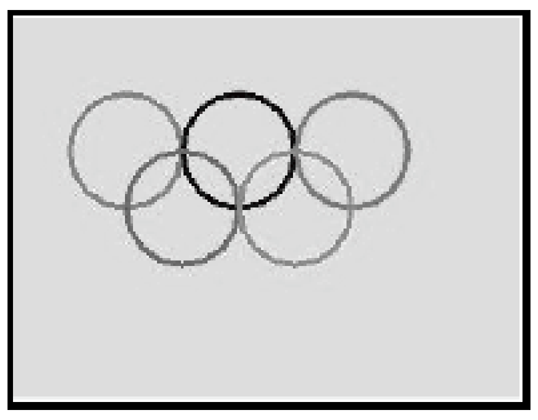


图 13.13　五环图案

关键技术

本实例主要是通过使用 Graphics2D 类的 setStroke()方法、setColor()方法和 drawOval()方法来实现的。

（1）使用 Graphics2D 类的 setStroke()方法指定笔画的粗细。

（2）使用 Graphics2D 类的 setColor()方法指定颜色。

（3）使用 Graphics2D 类的 drawOval()方法在指定位置绘制圆环，该方法是从 Graphics 类中继承的。

设计过程

（1）创建用于生成五环图案的 Servlet 类 DrawOvalServlet，然后在其 service()方法中实现绘制五环图案，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //禁止页面缓存
        response.setHeader("Pragma", "No-cache");
        response.setHeader("Cache-Control", "No-cache");
        response.setDateHeader("Expires", 0);
        response.setContentType("image/JPEG");                  //响应格式为 JPEG 图片
        //创建一个指定长宽的图像
        int width=270, height=200;
        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        Graphics g = image.getGraphics();                       //获取用于处理图形上下文的对象
        g.setColor(new Color(221,221,221));                     //设置第 1 个矩形的 RGB 颜色
        g.fillRect(0, 0, 270, 200);                             //绘制实心矩形
        Graphics2D g2 = (Graphics2D)g;                          //获得 Graphics2D 对象
        BasicStroke stroke = new BasicStroke(3);                //创建宽度是 3 的笔画对象
        g2.setStroke(stroke);                                   //设置笔画对象
        Color color = new Color(0,162,232);                     //创建颜色对象
        g2.setColor(color);                                     //设置颜色
        g2.drawOval(30, 40, 60, 60);                            //绘制第 1 个圆
        ……//此处省略了其他部分代码
        g.dispose();                                            //释放此图形的上下文以及它使用的所有系统资源
        ImageIO.write(image, "JPEG", response.getOutputStream()); //输出 JPEG 格式图片到页面
        response.getOutputStream().flush();                     //刷新输出流
        response.getOutputStream().close();                     //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页，在该页中添加一个<img>标签，调用 DrawOvalServlet 类生成图像并显示在页面中，关键代码如下：

```
<img alt="" src="DrawOvalServlet">
```

秘笈心法

在五环图案中，每种颜色都有特定的含义，为了获得五环图案的颜色，可以在 Photoshop 中按 F8 键，在打开的信息面板中获得颜色的 RGB 值，然后使用 Color 类的构造方法创建颜色对象。

实例 340	绘制艺术图案 光盘位置：光盘\MR\13\340	初级 实用指数：★★★☆

实例说明

本实例演示如何使用坐标轴平移、图形旋转和获得随机数等技术绘制艺术图案。运行程序，将在网页上显示出绘制的艺术图案，效果如图 13.14 所示。

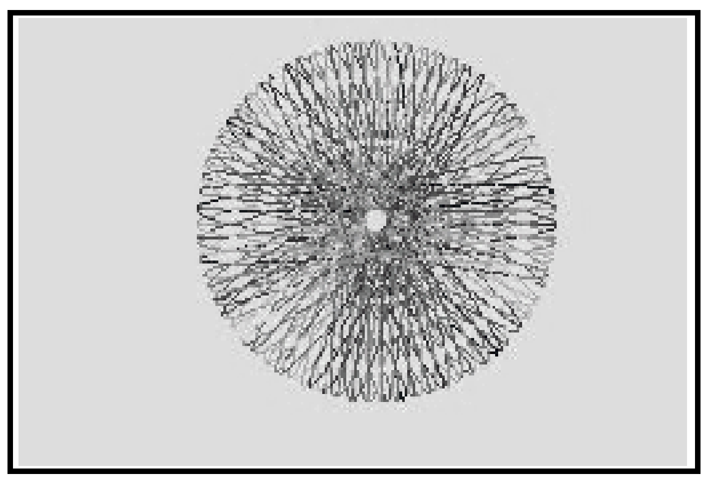


图 13.14　艺术图案

关键技术

本实例主要是使用 Graphics2D 类的 translate()方法、setColor()方法、rotate()方法和 draw()方法来实现的。

（1）使用 Graphics2D 类的 translate()方法将坐标轴平移到指定点。

（2）使用 Graphics2D 类的 setColor()方法设置颜色。

（3）使用 Graphics2D 类的 rotate()方法旋转绘图上下文。

（4）使用 Graphics2D 类的 draw()方法在指定位置绘制椭圆。

设计过程

（1）创建用于生成艺术图案的 Servlet 类 DrawArtServlet，然后在其 service()方法中实现绘制艺术图案，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
	//禁止页面缓存
	response.setHeader("Pragma", "No-cache");
	response.setHeader("Cache-Control", "No-cache");
	response.setDateHeader("Expires", 0);
	response.setContentType("image/JPEG");                    //响应格式为 JPEG 图片
	//创建一个指定长宽的图像
	int width=300, height=200;
	BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
	Graphics g = image.getGraphics();                         //获取用于处理图形上下文的对象
	g.setColor(new Color(221,221,221));                       //设置第 1 个矩形的 RGB 颜色
	g.fillRect(0, 0, 300, 200);                               //绘制实心矩形
	Graphics2D g2 = (Graphics2D)g;                            //获得 Graphics2D 对象
	Ellipse2D.Float ellipse = new Ellipse2D.Float(-80, 5, 160, 10);   //创建椭圆对象
	Random random = new Random();                             //创建随机数对象
	g2.translate(160, 90);                                    //平移坐标轴
	int R = random.nextInt(256);                              //随机产生颜色的 R 值
	int G = random.nextInt(256);                              //随机产生颜色的 G 值
	int B = random.nextInt(256);                              //随机产生颜色的 B 值
	Color color = new Color(R,G,B);                           //创建颜色对象
	g2.setColor(color);                                       //指定颜色
	g2.draw(ellipse);                                         //绘制椭圆
	int i=0;
	while (i<100){
		R = random.nextInt(256);                              //随机产生颜色的 R 值
		G = random.nextInt(256);                              //随机产生颜色的 G 值
		B = random.nextInt(256);                              //随机产生颜色的 B 值
		color = new Color(R,G,B);                             //创建新的颜色对象
		g2.setColor(color);                                   //指定颜色
		g2.rotate(10);                                        //旋转画布
		g2.draw(ellipse);                                     //绘制椭圆
		i++;
	}
	g2.dispose();
	g.dispose();                                              //释放此图形的上下文以及它使用的所有系统资源
	ImageIO.write(image, "JPEG", response.getOutputStream()); //输出 JPEG 格式图片到页面
	response.getOutputStream().flush();                       //刷新输出流
	response.getOutputStream().close();                       //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页，在该页中添加一个<img>标签，调用 DrawArtServlet 类生成图像并

显示在页面中，关键代码如下：

```
<img alt="" src="DrawArtServlet">
```

秘笈心法

使用 Random 类的实例生成伪随机数流，并使用该类的 nextInt(int n)方法，产生一个 0（包含）～n（不包含）之间的随机整数，由于颜色的 RGB 值是 0～255 之间的整数值，所以为 nextInt(int n)方法的参数 n 传递 256，这样就可以随机产生一个 0～255 之间的整数表示颜色 RGB 值。

实例 341　绘制花瓣

光盘位置：光盘\MR\13\341

初级

实用指数：★★★☆

实例说明

本实例演示如何使用坐标轴平移和图形旋转等技术绘制花瓣。运行程序，将在网页上显示绘制的花瓣，效果如图 13.15 所示。

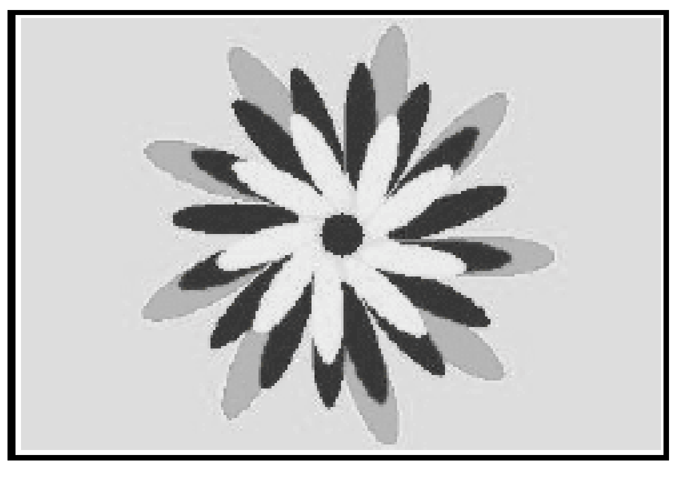


图 13.15　绘制花瓣

关键技术

本实例主要是通过使用 Graphics2D 类的 translate()方法、setColor()方法、rotate()方法和 fill()方法来实现的。

（1）使用 Graphics2D 类的 translate()方法将坐标轴平移到指定点。

（2）使用 Graphics2D 类的 setColor()方法设置颜色。

（3）使用 Graphics2D 类的 rotate()方法旋转绘图上下文。

（4）使用 Graphics2D 类的 fill()方法在指定位置绘制带填充色的椭圆。

设计过程

（1）创建用于生成花瓣图案的 Servlet 类 DrawFlowerServlet，然后在其 service()方法中实现绘制花瓣图案，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    //禁止页面缓存
    response.setHeader("Pragma", "No-cache");
    response.setHeader("Cache-Control", "No-cache");
    response.setDateHeader("Expires", 0);
    response.setContentType("image/JPEG");                //响应格式为 JPEG 图片
    //创建一个指定长宽的图像
    int width=300, height=200;
    BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
    Graphics g = image.getGraphics();                     //获取用于处理图形上下文的对象
    g.setColor(new Color(221,221,221));                   //设置第 1 个矩形的 RGB 颜色
    g.fillRect(0, 0, width, height);                      //绘制实心矩形
    Graphics2D g2 = (Graphics2D)g;                        //获得 Graphics2D 对象
    g2.translate(width/2,height/2);                       //平移坐标轴
    //绘制绿色花瓣
    Ellipse2D.Float ellipse = new Ellipse2D.Float(30, 0, 70, 20);  //创建椭圆对象
    Color color = new Color(0,255,0);                     //创建颜色对象
    g2.setColor(color);                                   //指定颜色
    g2.fill(ellipse);                                     //绘制椭圆
    int i=0;
    while (i<8){
        g2.rotate(30);                                    //旋转画布
        g2.fill(ellipse);                                 //绘制椭圆
        i++;
    }
    //绘制红色花瓣
```

```
        ellipse = new Ellipse2D.Float(20, 0, 60, 15);                //创建椭圆对象
        color = new Color(255,0,0);                                  //创建颜色对象
        g2.setColor(color);                                          //指定颜色
        g2.fill(ellipse);                                            //绘制椭圆
        i=0;
        while (i<15){
                g2.rotate(75);                                       //旋转画布
                g2.fill(ellipse);                                    //绘制椭圆
                i++;
        }
        //绘制黄色花瓣
        ellipse = new Ellipse2D.Float(10, 0, 50, 15);                //创建椭圆对象
        color = new Color(255,255,0);                                //创建颜色对象
        g2.setColor(color);                                          //指定颜色
        g2.fill(ellipse);                                            //绘制椭圆
        i=0;
        while (i<8){
                g2.rotate(30);                                       //旋转画布
                g2.fill(ellipse);                                    //绘制椭圆
                i++;
        }
        //绘制红色中心点
        color = new Color(255, 0, 0);                                //创建颜色对象
        g2.setColor(color);                                          //指定颜色
        ellipse = new Ellipse2D.Float(-10, -10, 20, 20);             //创建椭圆对象
        g2.fill(ellipse);
        g2.dispose();
        g.dispose();                                                 //释放此图形的上下文以及它使用的所有系统资源
        ImageIO.write(image, "JPEG", response.getOutputStream());    //输出 JPEG 格式图片到页面
        response.getOutputStream().flush();                          //刷新输出流
        response.getOutputStream().close();                          //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页，在该页中添加一个<img>标签，调用 DrawFlowerServlet 类生成图像并显示在页面中，关键代码如下：

```
<img alt="" src="DrawFlowerServlet">
```

秘笈心法

利用图形旋转技术和坐标轴平移，还可以实现时钟的绘制，方法是通过线程或定时器控件在指定的时间间隔内绕坐标轴分别旋转表示秒针、分针和时针的图形或图像，从而达到时钟显示时间的效果。

实例 342	绘制公章 光盘位置：光盘\MR\13\342	初级 实用指数：★★★☆

实例说明

本实例演示如何使用坐标轴平移和缩放、绘制椭圆、绘制多边形和绘制文本等技术实现公章的绘制。运行程序，将在网页上显示绘制的公章，效果如图 13.16 所示。



图 13.16　绘制公章

关键技术

本实例主要是使用 Graphics2D 类的 translate()、setColor()、scale()、drawString()、fillPolygon()和 draw()等方法来实现的。

（1）使用 Graphics2D 类的 translate()方法将坐标轴平移到指定点。

（2）使用 Graphics2D 类的 setColor()方法设置颜色。

（3）使用 Graphics2D 类的 scale()方法对公章中的文本进行缩放。

（4）使用 Graphics2D 类的 drawString()方法绘制文本，该方法是从 Graphics 类继承的。

（5）使用 Graphics2D 类的 fillPolygon()方法绘制公章的五星，该方法也是从 Graphics 类继承的。

（6）使用 Graphics2D 类的 draw()方法绘制表示公章的圆。

设计过程

（1）创建用于生成公章图案的 Servlet 类 DrawCachetServlet，然后在其 service()方法中实现绘制公章图案，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //禁止页面缓存
        response.setHeader("Pragma", "No-cache");
        response.setHeader("Cache-Control", "No-cache");
        response.setDateHeader("Expires", 0);
        response.setContentType("image/JPEG");                    //响应格式为 JPEG 图片
        //创建一个指定长宽的图像
        int width=325, height=205;
        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        Graphics g = image.getGraphics();                         //获取用于处理图形上下文的对象
        g.setColor(new Color(221,221,221));                       //设置第 1 个矩形的 RGB 颜色
        g.fillRect(0, 0, width, height);                          //绘制实心矩形
        Graphics2D g2 = (Graphics2D) g;                           //获得 Graphics2D 对象
        g2.translate(170, 100);                                   //平移坐标轴
        BasicStroke stroke = new BasicStroke(6);                  //创建宽度是 6 的笔画对象
        g2.setStroke(stroke);                                     //设置笔画对象
        //绘制圆
        Ellipse2D.Float ellipse = new Ellipse2D.Float(-80, -80, 160, 160);     //创建圆对象
        Color color = new Color(255, 0, 0);                       //创建颜色对象
        g2.setColor(color);                                       //指定颜色
        g2.draw(ellipse);                                         //绘制圆
        //绘制五星
        int[] x1 = { 0, 8, 30, 16, 25, 0, -25, -16, -30, -8 };    //多边形的横坐标
        int[] y1 = { -35, -10, -15, 5, 28, 10, 28, 5, -15, -10 }; //多边形的纵坐标
        int n1 = 10;                                              //多边形的边数
        g2.fillPolygon(x1, y1, n1);                               //绘制多边形
        //绘制文本
        g2.scale(1.8, 1.8);                                       //放大
        Font font = new Font("宋体", Font.BOLD, 12);              //创建字体
        g2.setFont(font);                                         //设置字体
        g2.drawString("专  用  章", -25, 30);                     //绘制文本
        char[] array = "  明日科技有限公司              ".toCharArray();    //把字符串转换为字符数组
        int len = array.length * 2;                               //定义半径
        font = new Font("宋体", Font.BOLD, 10);                   //创建新字体
        g2.setFont(font);                                         //设置字体
        double angle = 0;                                         //初始角度
        for (int i = 0; i < array.length; i++) {                  //遍历字符串中的字符
                int x = (int) (len * Math.sin(Math.toRadians(angle + 270)));    //计算每个文字的位置
                int y = (int) (len * Math.cos(Math.toRadians(angle + 270)));    //计算每个文字的位置
                g2.drawString(array[i] + "", width / 2 + x - 168, height / 2 - y - 95);  //绘制每个字符，其中 168 和 95 是坐标平移值
                angle = angle + 360d / array.length;              //改变角度
        }
        g2.dispose();
        g.dispose();                                              //释放此图形的上下文以及它使用的所有系统资源
        ImageIO.write(image, "JPEG", response.getOutputStream()); //输出 JPEG 格式图片到页面
        response.getOutputStream().flush();                       //刷新输出流
        response.getOutputStream().close();                       //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页，在该页中添加一个<img>标签，调用 DrawCachetServlet 类生成图像并显示在页面中，关键代码如下：

```
<img alt="" src="DrawCachetServlet">
```

秘笈心法

在进行图形和文本的绘制时，要求能够正确设置绘图上下文的属性，如文本的字体、颜色、图形线条的粗

细、虚线还是实线，以及颜色等，正确设置的方法是在绘制每一种新样式的文本或图形之前，先对绘图上下文的属性进行设置，然后再绘制文本和图形，这样设置的绘图上下文属性才是有效的。

13.3 图形的合并运算

实例 343 图形的加运算

光盘位置：光盘\MR\13\343

初级

实用指数：★★★☆

实例说明

本实例演示在 Java 中如何实现图形的加运算，即取两个图形的并集。运行程序，将在网页上显示进行加运算后的图形，效果如图 13.17 所示。

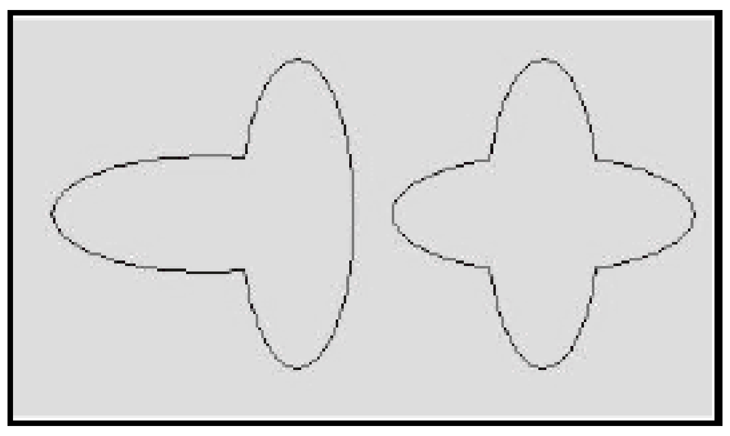


图 13.17 图形进行加运算的效果

关键技术

本实例主要是通过 Graphics2D 类的 draw()方法和 Area 类来实现的，其中 Area 类用于封装图形对象，并通过 add()方法对封装的图形对象进行加运算。

（1）使用 Area 类的构造方法封装图形对象，其构造方法的定义如下：

```
public Area(Shape s)
```

参数说明

s：是 Area 类封装的图形对象。

（2）使用 Area 类的 add()方法对封装的图形对象进行加运算，该方法的定义如下：

参数说明

```
public void add(Area rhs)
```

rhs：与当前 Area 对象进行加运算的 Area 对象。

设计过程

（1）创建用于进行图形加运算的 Servlet 类 PlusOperationServlet，然后在其 service()方法中实现图形的加运算，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //禁止页面缓存
        response.setHeader("Pragma", "No-cache");
        response.setHeader("Cache-Control", "No-cache");
        response.setDateHeader("Expires", 0);
        response.setContentType("image/JPEG");                          //响应格式为 JPEG 图片
        //创建一个指定长宽的图像
        int width=370, height=205;
        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        Graphics g = image.getGraphics();                               //获取用于处理图形上下文的对象
        g.setColor(new Color(221,221,221));                             //设置第 1 个矩形的 RGB 颜色
        g.fillRect(0, 0, width, height);                                //绘制实心矩形
        Graphics2D g2 = (Graphics2D)g;                                  //获得 Graphics2D 对象
        g2.setColor(Color.BLUE);
        Ellipse2D.Float ellipse1 = new Ellipse2D.Float(20, 70, 160, 60);    //创建椭圆对象
        Ellipse2D.Float ellipse2 = new Ellipse2D.Float(120, 20, 60, 160);   //创建椭圆对象
        Area area1 = new Area(ellipse1);                                //创建区域椭圆
        Area area2 = new Area(ellipse2);                                //创建区域椭圆
        area1.add(area2);                                               //两个区域椭圆进行加运算
        g2.draw(area1);                                                 //绘制加运算后的区域椭圆
        Ellipse2D.Float ellipse3 = new Ellipse2D.Float(200, 70, 160, 60);   //创建椭圆对象
        Ellipse2D.Float ellipse4 = new Ellipse2D.Float(250, 20, 60, 160);   //创建椭圆对象
```

```
        Area area3 = new Area(ellipse3);                                    //创建区域椭圆
        Area area4 = new Area(ellipse4);                                    //创建区域椭圆
        area3.add(area4);                                                   //两个区域椭圆进行加运算
        g2.draw(area3);                                                     //绘制加运算后的区域椭圆
        g2.dispose();
        g.dispose();                                                        //释放此图形的上下文以及它使用的所有系统资源
        ImageIO.write(image, "JPEG", response.getOutputStream());           //输出 JPEG 格式图片到页面
        response.getOutputStream().flush();                                 //刷新输出流
        response.getOutputStream().close();                                 //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页，在该页中添加一个<img>标签，调用 PlusOperationServlet 类生成图像并显示在页面中，关键代码如下：

```
<img alt="" src="PlusOperationServlet">
```

秘笈心法

在调用完 Graphics 或 Graphics2D 之后，一定不要忘记调用其 dispose()方法，释放图形上下文以及它使用的所有系统资源。

实例 344	图形的减运算 光盘位置：光盘\MR\13\344	初级 实用指数：★★★☆

实例说明

本实例演示在 Java 中如何实现图形的减运算，即从当前图形中减去与另一个图形的交集。运行程序，将在网页上显示进行减运算后的图形，效果如图 13.18 所示。

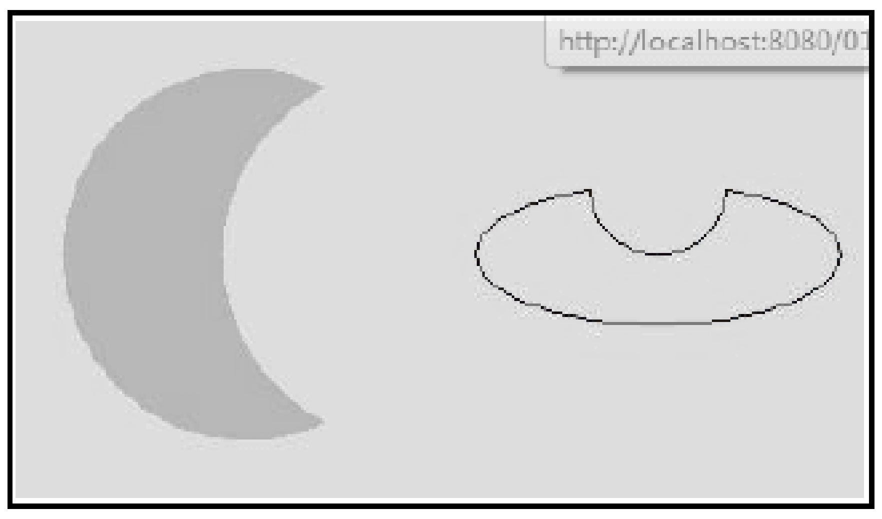


图 13.18　图形进行减运算的效果

关键技术

本实例主要是使用 Graphics2D 类的 draw()方法和 Area 类来实现的，其中 Area 类用于封装图形对象，并通过 subtract()方法对封装的图形对象进行减运算。

使用 Area 类的 subtract()方法对封装的图形对象进行减运算，该方法的定义如下：

```
public void subtract(Area rhs)
```

参数说明

rhs：与当前 Area 对象进行减运算的 Area 对象。

设计过程

（1）创建用于进行图形减运算的 Servlet 类 SubtractOperationServlet，然后在其 service()方法中实现图形的减运算，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //禁止页面缓存
        response.setHeader("Pragma", "No-cache");
        response.setHeader("Cache-Control", "No-cache");
        response.setDateHeader("Expires", 0);
        response.setContentType("image/JPEG");                              //响应格式为 JPEG 图片
        //创建一个指定长宽的图像
        int width=370, height=205;
        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        Graphics g = image.getGraphics();                                   //获取用于处理图形上下文的对象
        g.setColor(new Color(221,221,221));                                 //设置第 1 个矩形的 RGB 颜色
        g.fillRect(0, 0, width, height);                                    //绘制实心矩形
        Graphics2D g2 = (Graphics2D)g;                                      //获得 Graphics2D 对象
        Ellipse2D.Float ellipse1 = new Ellipse2D.Float(20, 20, 160, 160);   //创建圆对象
```

```
        Ellipse2D.Float ellipse2 = new Ellipse2D.Float(90, 20, 160, 160);        //创建圆对象
        Area area1 = new Area(ellipse1);                                         //创建区域圆
        Area area2 = new Area(ellipse2);                                         //创建区域圆
        area1.subtract(area2);                                                   //两个区域圆进行减运算
        g2.setColor(new Color(178,186,174));
        g2.fill(area1);                                                          //绘制减运算后的区域圆
        Ellipse2D.Float ellipse3 = new Ellipse2D.Float(200, 70, 160, 60);        //创建椭圆对象
        Ellipse2D.Float ellipse4 = new Ellipse2D.Float(250, 40, 60, 60);         //创建圆对象
        Area area3 = new Area(ellipse3);                                         //创建区域椭圆
        Area area4 = new Area(ellipse4);                                         //创建区域圆
        area3.subtract(area4);                                                   //两个区域图形进行减运算
        g2.setColor(Color.BLUE);
        g2.draw(area3);                                                          //绘制减运算后的区域图形
        g2.dispose();
        g.dispose();                                                             //释放此图形的上下文以及它使用的所有系统资源
        ImageIO.write(image, "JPEG", response.getOutputStream());                //输出 JPEG 格式图片到页面
        response.getOutputStream().flush();                                      //刷新输出流
        response.getOutputStream().close();                                      //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页，在该页中添加一个<img>标签，调用 SubtractOperationServlet 类生成图像并显示在页面中，关键代码如下：

```
<img alt="" src="SubtractOperationServlet">
```

秘笈心法

在利用 Servlet 生成图像时，必须设置响应正文的 MIME 类型为图片格式 image/jpeg，这样才能生成一个图片。

实例 345 图形的交运算

光盘位置：光盘\MR\13\345

初级

实用指数：★★★★☆

实例说明

本实例演示在 Java 中如何实现图形的交运算，即保留两个图形的交集。运行程序，将在网页上显示进行交运算后的图形，效果如图 13.19 所示。

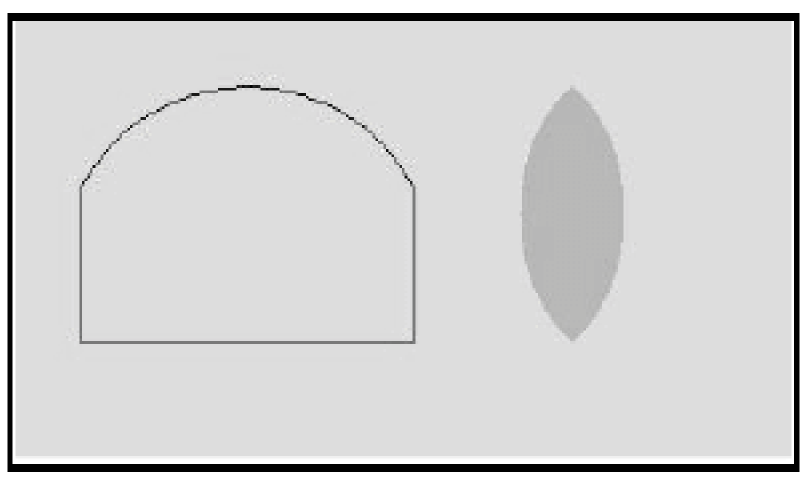


图 13.19　图形进行交运算的效果

关键技术

本实例主要是使用 Graphics2D 类的 draw()方法和 Area 类来实现的，其中 Area 类用于封装图形对象，并通过 intersect()方法对封装的图形对象进行交运算。

使用 Area 类的 intersect()方法对封装的图形对象进行交运算，该方法的定义如下：

```
public void intersect(Area rhs)
```

参数说明

rhs：与当前 Area 对象进行交运算的 Area 对象。

设计过程

（1）创建用于进行图形交运算的 Servlet 类 IntersectOperationServlet，然后在其 service()方法中实现图形的交运算，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //禁止页面缓存
        response.setHeader("Pragma", "No-cache");
        response.setHeader("Cache-Control", "No-cache");
        response.setDateHeader("Expires", 0);
```

```
        response.setContentType("image/JPEG");                    //响应格式为 JPEG 图片
        //创建一个指定长宽的图像
        int width=370, height=205;
        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        Graphics g = image.getGraphics();                          //获取用于处理图形上下文的对象
        g.setColor(new Color(221,221,221));                        //设置第 1 个矩形的 RGB 颜色
        g.fillRect(0, 0, width, height);                           //绘制实心矩形
        Graphics2D g2 = (Graphics2D)g;                             //获得 Graphics2D 对象
        g2.setColor(Color.BLUE);
        Rectangle2D.Float rect = new Rectangle2D.Float(30, 30, 160, 120);  //创建矩形对象
        Ellipse2D.Float ellipse = new Ellipse2D.Float(20, 30, 180, 180);   //创建圆对象
        Area area1 = new Area(rect);                               //创建区域矩形
        Area area2 = new Area(ellipse);                            //创建区域圆
        area1.intersect(area2);                                    //两个区域图形进行交运算
        g2.draw(area1);                                            //绘制交运算后的区域图形
        Ellipse2D.Float ellipse1 = new Ellipse2D.Float(190, 20, 100, 140); //创建椭圆对象
        Ellipse2D.Float ellipse2 = new Ellipse2D.Float(240, 20, 100, 140); //创建椭圆对象
        Area area3 = new Area(ellipse1);                           //创建区域椭圆
        Area area4 = new Area(ellipse2);                           //创建区域椭圆
        area3.intersect(area4);                                    //两个区域椭圆进行交运算
        g2.setColor(new Color(178,186,174));
        g2.fill(area3);                                            //绘制交运算后的区域椭圆
        g2.dispose();
        g.dispose();                                               //释放此图形的上下文以及它使用的所有系统资源
        ImageIO.write(image, "JPEG", response.getOutputStream());  //输出 JPEG 格式图片到页面
        response.getOutputStream().flush();                        //刷新输出流
        response.getOutputStream().close();                        //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页，在该页中添加一个<img>标签，调用 IntersectOperationServlet 类生成图像并显示在页面中，关键代码如下：

```
<img alt="" src="IntersectOperationServlet">
```

秘笈心法

利用 Area 类的交运算方法 intersect()可以轻松地获取两个不同形状的图形的交集，而用户只需要创建不同形状的图形对象，然后调用其 intersect()即可，并不需要编写大量代码计算它是如何进行交运算的。

实例 346	图形的异或运算 光盘位置：光盘\MR\13\346	初级 实用指数：★★★☆

实例说明

本实例演示在 Java 中如何实现图形的异或运算，即两个图形去除交集后剩下的部分。运行程序，将在网页上显示进行异或运算后的图形，效果如图 13.20 所示。

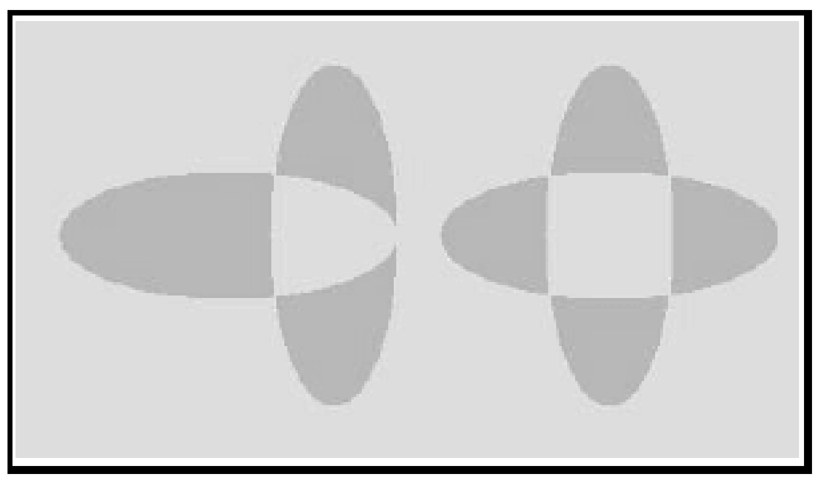


图 13.20　图形进行异或运算的效果

关键技术

本实例主要是使用 Graphics2D 类的 draw()方法和 Area 类来实现的，其中 Area 类用于封装图形对象，并通过 exclusiveOr()方法对封装的图形对象进行异或运算。

使用 Area 类的 exclusiveOr()方法对封装的图形对象进行异或运算，该方法的定义如下：

```
public void exclusiveOr(Area rhs)
```

参数说明

rhs：与当前 Area 对象进行异或运算的 Area 对象。

设计过程

（1）创建用于进行图形异或运算的 Servlet 类 ExclusiveOrOperationServlet，然后在其 service()方法中实现图形的异或运算，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    //禁止页面缓存
    response.setHeader("Pragma", "No-cache");
    response.setHeader("Cache-Control", "No-cache");
    response.setDateHeader("Expires", 0);
    response.setContentType("image/JPEG");                    //响应格式为 JPEG 图片
    //创建一个指定长宽的图像
    int width=370, height=205;
    BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
    Graphics g = image.getGraphics();                         //获取用于处理图形上下文的对象
    g.setColor(new Color(221,221,221));                       //设置第 1 个矩形的 RGB 颜色
    g.fillRect(0, 0, width, height);                          //绘制实心矩形
    Graphics2D g2 = (Graphics2D)g;                            //获得 Graphics2D 对象
    g2.setColor(new Color(178,186,174));
    Ellipse2D.Float ellipse1 = new Ellipse2D.Float(20, 70, 160, 60);     //创建椭圆对象
    Ellipse2D.Float ellipse2 = new Ellipse2D.Float(120, 20, 60, 160);    //创建椭圆对象
    Area area1 = new Area(ellipse1);                          //创建区域椭圆
    Area area2 = new Area(ellipse2);                          //创建区域椭圆
    area1.exclusiveOr(area2);                                 //两个区域椭圆进行异或运算
    g2.fill(area1);                                           //绘制异或运算后的区域椭圆
    Ellipse2D.Float ellipse3 = new Ellipse2D.Float(200, 70, 160, 60);    //创建椭圆对象
    Ellipse2D.Float ellipse4 = new Ellipse2D.Float(250, 20, 60, 160);    //创建椭圆对象
    Area area3 = new Area(ellipse3);                          //创建区域椭圆
    Area area4 = new Area(ellipse4);                          //创建区域椭圆
    area3.exclusiveOr(area4);                                 //两个区域椭圆进行异或运算
    g2.fill(area3);                                           //绘制异或运算后的区域椭圆
    g2.dispose();
    g.dispose();                                              //释放此图形的上下文以及它使用的所有系统资源
    ImageIO.write(image, "JPEG", response.getOutputStream()); //输出 JPEG 格式图片到页面
    response.getOutputStream().flush();                       //刷新输出流
    response.getOutputStream().close();                       //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页，在该页中添加一个<img>标签，调用 ExclusiveOrOperationServlet 类生成图像并显示在页面中，关键代码如下：

```
<img alt="" src="ExclusiveOrOperationServlet">
```

秘笈心法

利用 Area 类的异或运算方法 exclusiveOr()，可以轻松地对不同形状的图形进行异或运算，而用户只需要创建不同形状的图形对象，然后调用其 exclusiveOr()即可，并不需要编写大量代码计算它是如何进行异或运算的。

13.4 文 字 特 效

实例 347	立体效果的文字 光盘位置：光盘\MR\13\347	初级 实用指数：★★★☆

实例说明

本实例演示如何利用 Java 的绘图技术，实现立体效果文字的绘制。运行程序，将在网页上显示出绘制立体效果的文字，效果如图 13.21 所示。



图 13.21　立体效果的文字

关键技术

使用 Graphics 类的 setFont()方法设置完字体、字型和字号后，使用 Graphics 类的 setColor()方法将绘图上下文的前景色设置为灰色，然后使用 Graphics 类的 drawString()方法绘制文本，并且每次绘制的文本都向右向下移动一小段距离，最后再将绘图上下文的前景色更改为黑色，然后再绘制一次文本，从而实现立体字效果。

（1）使用 Graphics 类的 setFont()方法，并将 Font 类创建的字体对象作为 setFont()方法的参数，实现为文本设置字体的操作，setFont()方法的定义如下：

```
public abstract void setFont(Font font)
```

参数说明

font：为文本设置的字体对象。

（2）使用 Graphics 类的 setColor()方法，并将 Color 类创建的颜色对象作为 setColor()方法的参数，实现为文本和图形设置颜色的操作，setColor()方法的定义如下：

```
public abstract void setColor(Color color)
```

参数说明

color：为文本或图形设置的颜色对象。

（3）使用 Graphics 类的 drawString()方法绘制文本，该方法的定义如下：

```
public abstract void drawString(String str, int x, int y)
```

参数说明

❶ str：绘制的文本内容。

❷ x：绘制点的 x 坐标。

❸ y：绘制点的 y 坐标。

设计过程

（1）创建用于绘制立体效果文字的 Servlet 类 DrawSolidTextServlet，然后在其 service()方法中实现绘制立体效果的文字，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    //禁止页面缓存
    response.setHeader("Pragma", "No-cache");
    response.setHeader("Cache-Control", "No-cache");
    response.setDateHeader("Expires", 0);
    response.setContentType("image/JPEG");                  //响应格式为 JPEG 图片
    //创建一个指定长宽的图像
    int width=370, height=205;
    BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
    Graphics g = image.getGraphics();                        //获取用于处理图形上下文的对象
    g.setColor(new Color(221,221,221));                      //设置第 1 个矩形的 RGB 颜色
    g.fillRect(0, 0, width, height);                         //绘制实心矩形
    String value = "明日科技";
    int x = 16;                                              //文本位置的横坐标
    int y = 100;                                             //文本位置的纵坐标
    Font font = new Font("宋体", Font.BOLD, 72);             //创建字体对象
    g.setFont(font);                                         //设置字体
    g.setColor(Color.GRAY);                                  //设置颜色为灰色
    int i = 0;                                               //循环变量
    while (i<8){
        g.drawString(value, x, y);                           //绘制文本
```

```
            x+=1;                                            //调整绘制点的横坐标值
            y+=1;                                            //调整绘制点的纵坐标值
            i++;                                             //调整循环变量的值
        }
        g.setColor(Color.BLACK);                             //设置颜色为黑色
        g.drawString(value, x, y);                           //绘制文本
        g.dispose();                                         //释放此图形的上下文以及它使用的所有系统资源
        ImageIO.write(image, "JPEG", response.getOutputStream()); //输出 JPEG 格式图片到页面
        response.getOutputStream().flush();                  //刷新输出流
        response.getOutputStream().close();                  //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页，在该页中添加一个<img>标签，调用 DrawSolidTextServlet 类生成图像并显示在页面中，关键代码如下：

```
<img alt="" src="DrawSolidTextServlet">
```

秘笈心法

在程序中使用 Graphics 类的 drawString()方法绘制文本时，可以通过变量来指定文本绘制点的横、纵坐标，并在绘制文本时通过线程来改变文本绘制点的横、纵坐标，从而可以实现文本移动的动画效果。

实例 348	阴影效果的文字 光盘位置：光盘\MR\13\348	初级 实用指数：★★★☆

实例说明

本实例演示如何利用 Java 的绘图技术，实现阴影效果文字的绘制。运行程序，将在网页上显示阴影效果的文字，效果如图 13.22 所示。



图 13.22　阴影效果的文字

关键技术

使用 Graphics 类的 setFont()方法设置完字体、字型和字号后，使用 Graphics 类的 setColor()方法将绘图上下文的前景色设置为灰色，然后使用 Graphics 类的 drawString()方法绘制文本，然后再将绘图上下文的前景色更改为黑色，并且将绘制的文本都向左向上移动一小段距离，从而实现阴影文字的效果。

（1）使用 Graphics 类的 setFont()方法，并将 Font 类创建的字体对象作为 setFont()方法的参数，实现为文本设置字体的操作。

（2）使用 Graphics 类的 setColor()方法，并将 Color 类创建的颜色对象作为 setColor()方法的参数，实现为文本和图形设置颜色的操作。

（3）使用 Graphics 类的 drawString()方法绘制文本。

设计过程

（1）创建用于绘制阴影效果文字的 Servlet 类 DrawShadowTextServlet，然后在其 service()方法中实现绘制阴影效果的文字，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        //禁止页面缓存
        response.setHeader("Pragma", "No-cache");
        response.setHeader("Cache-Control", "No-cache");
        response.setDateHeader("Expires", 0);
        response.setContentType("image/JPEG");               //响应格式为 JPEG 图片
        //创建一个指定长宽的图像
        int width=370, height=205;
        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        Graphics g = image.getGraphics();                    //获取用于处理图形上下文的对象
```

```
        g.setColor(new Color(221,221,221));                         //设置第 1 个矩形的 RGB 颜色
        g.fillRect(0, 0, width, height);                            //绘制实心矩形
        String value = "编程词典";
        int x = 16;                                                 //文本位置的横坐标
        int y = 100;                                                //文本位置的纵坐标
        Font font = new Font("华文行楷", Font.BOLD, 72);             //创建字体对象
        g.setFont(font);                                            //设置字体
        g.setColor(Color.GRAY);                                     //设置颜色为灰色
        int i = 0;                                                  //循环变量
        g.drawString(value, x, y);                                  //绘制文本
        x -= 3;                                                     //调整绘制点的横坐标值
        y -= 3;                                                     //调整绘制点的纵坐标值
        g.setColor(Color.BLACK);                                    //设置颜色为黑色
        g.drawString(value, x, y);                                  //绘制文本
        g.dispose();                                                //释放此图形的上下文以及它使用的所有系统资源
        ImageIO.write(image, "JPEG", response.getOutputStream());   //输出 JPEG 格式图片到页面
        response.getOutputStream().flush();                         //刷新输出流
        response.getOutputStream().close();                         //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页，在该页中添加一个<img>标签，调用 DrawShadowTextServlet 类生成图像并显示在页面中，关键代码如下：

```
<img alt="" src="DrawShadowTextServlet">
```

秘笈心法

在绘制阴影文字时，有时会出现阴影不在文字的下方，这是由于先绘制的内容是显示在下面的，而后绘制的内容是显示在上面的，所以解决的方法是在绘制阴影文字时，应先将阴影绘制到绘图上下文上，然后再绘制需要显示的文本，并在横向和纵向分别偏移一小段距离，这样就能正常显示文字的阴影。

实例 349　倾斜效果的文字

光盘位置：光盘\MR\13\349

初级

实用指数：★★★☆

实例说明

本实例演示如何利用 Java 的绘图技术，实现倾斜效果文字的绘制。运行程序，将在网页上显示出倾斜效果的文字，效果如图 13.23 所示。



图 13.23　倾斜效果的文字

关键技术

使用 Graphics2D 类的 setShear()方法，倾斜绘图上下文对象，然后使用从 Graphics 类继承的 setFont()方法设置字体、字型和字号，使用 drawString()方法绘制文本，从而实现文字的倾斜效果。

（1）使用从 Graphics 类继承的 setFont()方法设置绘图上下文的字体。

（2）使用 Graphics2D 类的 setShear()方法，使绘图上下文倾斜，这样绘制的文本就是倾斜的文本，该方法的定义如下：

```
public abstract void shear(double shx, double shy)
```

参数说明

❶ shx：在正 X 轴方向移动坐标的乘数，它可以作为其 y 坐标的函数。

❷ shy：在正 Y 轴方向移动坐标的乘数，它可以作为其 x 坐标的函数。

（3）使用从 Graphics 类继承的 drawString()方法绘制文本。

设计过程

（1）创建用于绘制倾斜效果文字的 Servlet 类 DrawShearTextServlet，然后在其 service()方法中实现绘制倾

斜效果的文字，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    //禁止页面缓存
    response.setHeader("Pragma", "No-cache");
    response.setHeader("Cache-Control", "No-cache");
    response.setDateHeader("Expires", 0);
    response.setContentType("image/JPEG");                    //响应格式为 JPEG 图片
    //创建一个指定长宽的图像
    int width=370, height=205;
    BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
    Graphics g = image.getGraphics();                         //获取用于处理图形上下文的对象
    g.setColor(new Color(221,221,221));                       //设置第 1 个矩形的 RGB 颜色
    g.fillRect(0, 0, width, height);                          //绘制实心矩形
    Graphics2D g2 = (Graphics2D)g;                            //转换为 Graphics2D 类型
    g2.setColor(Color.BLACK);
    String value = "编程词典";                                 //绘制的文本
    int x = 10;                                               //文本位置的横坐标
    int y = 170;                                              //文本位置的纵坐标
    Font font = new Font("华文行楷", Font.BOLD, 72);           //创建字体对象
    g2.setFont(font);                                         //设置字体
    g2.shear(0.1, -0.4);                                      //倾斜画布
    g2.drawString(value, x, y);                               //绘制文本
    g2.dispose();
    g.dispose();                                              //释放此图形的上下文以及它使用的所有系统资源
    ImageIO.write(image, "JPEG", response.getOutputStream()); //输出 JPEG 格式图片到页面
    response.getOutputStream().flush();                       //刷新输出流
    response.getOutputStream().close();                       //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页，在该页中添加一个<img>标签，调用 DrawShearTextServlet 类生成图像并显示在页面中，关键代码如下：

```
<img alt="" src="DrawShearTextServlet">
```

秘笈心法

在 Java 中可以通过 Polygon 类创建多边形，从而实现平行四边形的绘制，但是使用该类有一个缺点，就是必须要正确计算出每个点的横、纵坐标，为此可以使用 Graphics2D 类的 rotate()方法旋转绘图上下文，并使用 shear()方法倾斜绘图上下文，然后再绘制矩形，这样所绘制的矩形就可以显示为平行四边形。

实例 350	渐变效果的文字 光盘位置：光盘\MR\13\350	初级 实用指数：★★★☆

实例说明

在程序中绘制的文本信息通常都是单一的颜色，本实例使用 GradientPaint 类创建封装渐变颜色的对象，并为绘图上下文指定该对象，从而实现绘制渐变效果文字的功能。运行程序，将在网页上显示出绘制渐变效果的文字，效果如图 13.24 所示。

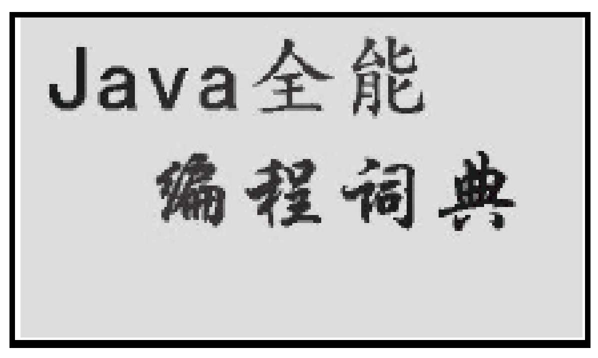


图 13.24　渐变效果的文字

关键技术

本实例主要是使用 Graphics2D 类的 setPaint()方法来实现的，为绘图上下文指定 GradientPaint 类创建的渐变色对象，从而实现绘制渐变效果文字的功能。

（1）使用 Graphics2D 类的 setPaint()方法，并将 GradientPaint 类创建的封装渐变颜色的对象作为 setPaint()方法的参数，实现为图形填充渐变色的操作，setPaint()方法的定义如下：

```
public abstract void setPaint(Paint paint)
```

参数说明

paint：封装了渐变颜色的 Paint 对象。

（2）使用 GradientPaint 类创建封装渐变颜色的对象，其构造方法的定义如下：

```
public GradientPaint(float x1, float y1, Color color1, float x2, float y2, Color color2, boolean cyclic)
```

参数说明

❶ x1：用户空间中第一个指定点的 x 坐标。

❷ y1：用户空间中第一个指定点的 y 坐标。

❸ color1：第一个指定点处的 Color 对象。

❹ x2：用户空间中第二个指定点的 x 坐标。

❺ y2：用户空间中第二个指定点的 y 坐标。

❻ color2：第二个指定点处的 Color 对象。

❼ cyclic：如果渐变模式在两种颜色之间重复循环，则该值设置为 true；否则设置为 false。

设计过程

（1）创建用于绘制渐变效果文字的 Servlet 类 DrawGradientTextServlet，然后在其 service()方法中实现绘制渐变效果的文字，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    //禁止页面缓存
    response.setHeader("Pragma", "No-cache");
    response.setHeader("Cache-Control", "No-cache");
    response.setDateHeader("Expires", 0);
    response.setContentType("image/JPEG");                    //响应格式为 JPEG 图片
    //创建一个指定长宽的图像
    int width=370, height=205;
    BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
    Graphics g = image.getGraphics();                          //获取用于处理图形上下文的对象
    g.setColor(new Color(221,221,221));                        //设置第 1 个矩形的 RGB 颜色
    g.fillRect(0, 0, width, height);                           //绘制实心矩形
    Graphics2D g2 = (Graphics2D)g;                             //转换为 Graphics2D 类型
    String value = "Java 全能";                                //绘制的文本
    int x = 15;                                                //文本位置的横坐标
    int y = 60;                                                //文本位置的纵坐标
    Font font = new Font("楷体", Font.BOLD, 60);               //创建字体对象
    g2.setFont(font);                                          //设置字体
    //创建循环渐变的 GradientPaint 对象
    GradientPaint paint = new GradientPaint(20, 20, Color.BLUE, 100,120, Color.RED, true);
    g2.setPaint(paint);                                        //设置渐变
    g2.drawString(value, x, y);                                //绘制文本
    font = new Font("华文行楷", Font.BOLD, 60);                //创建新的字体对象
    g2.setFont(font);                                          //设置字体
    x = 80;                                                    //文本位置的横坐标
    y = 130;                                                   //文本位置的纵坐标
    value = "编程词典";                                        //绘制的文本
    g2.drawString(value, x, y);                                //绘制文本
    g2.dispose();
    g.dispose();                                               //释放此图形的上下文以及它使用的所有系统资源
    ImageIO.write(image, "JPEG", response.getOutputStream()); //输出 JPEG 格式图片到页面
    response.getOutputStream().flush();                        //刷新输出流
    response.getOutputStream().close();                        //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页，在该页中添加一个<img>标签，调用 DrawGradientTextServlet 类生成图像并显示在页面中，关键代码如下：

```
<img alt="" src="DrawGradientTextServlet">
```

秘笈心法

在绘制具有渐变颜色的图形和文本时，如果对各种颜色的距离有所限制，可以在使用 GradientPaint 类创建渐变对象时，通过改变起始点和终止点之间的距离来实现。

实例 351 水印文字特效

光盘位置：光盘\MR\13\351

初级

实用指数：★★★☆

实例说明

水印文字是通过改变绘图上下文的透明度来实现的，本实例将演示水印文字的实现。运行程序，将在网页上显示绘制图片，并为图片添加水印文字，效果如图 13.25 所示。



图 13.25　水印文字特效

关键技术

本实例主要是通过 Graphics2D 类的 setComposite()方法，为绘图上下文指定表示透明度的 AlphaComposite 对象实现的。

（1）使用 AlphaComposite 类获得表示透明度的 AlphaComposite 对象，该对象使用 AlphaComposite 类的字段 SrcOver 调用 derive()方法获得，该方法的定义如下：

```
public AlphaComposite derive(float alpha)
```

参数说明

❶ alpha：闭区间 0.0f～1.0f 之间的一个浮点数字，为 0.0f 时完全透明，为 1.0f 时不透明。

❷ 返回值：表示透明度的 AlphaComposite 对象。

（2）使用 Graphics2D 类的 setComposite()方法，为绘图上下文指定表示透明度的 AlphaComposite 对象，该方法的定义如下：

```
public abstract void setComposite(Composite comp)
```

参数说明

comp：表示透明度的 AlphaComposite 对象。

设计过程

（1）创建用于绘制图片并为图片添加水印效果文字的 Servlet 类 DrawWatermarkTextServlet，然后在其 service()方法中实现绘制图片并为图片添加水印效果，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        //禁止页面缓存
        response.setHeader("Pragma", "No-cache");
        response.setHeader("Cache-Control", "No-cache");
        response.setDateHeader("Expires", 0);
        response.setContentType("image/JPEG");                                  //响应格式为 JPEG 图片
        //创建一个指定长宽的图像
        int width=470, height=305;
        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        Graphics g = image.getGraphics();                                       //获取用于处理图形上下文的对象
        g.setColor(new Color(221,221,221));                                     //设置第 1 个矩形的 RGB 颜色
        g.fillRect(0, 0, width, height);                                        //绘制实心矩形
        Graphics2D g2 = (Graphics2D)g;                                          //转换为 Graphics2D 类型
        g2.setColor(Color.BLACK);
        String imgPath = getServletContext().getRealPath("/img/image.jpg");     //获取图片的路径
        BufferedImage bufferImg = ImageIO.read(new File(imgPath));              //读取图片数据，转换为 BufferImage 对象
        AffineTransformOp op = new AffineTransformOp(new AffineTransform(),null) ; //根据仿射转换和插值类型构造一个 AffineTransformOp
        g2.drawImage(bufferImg,op, 0, 0);                                       //绘制图像
        g2.rotate(Math.toRadians(-30));                                         //旋转绘图上下文对象
        Font font = new Font("楷体",Font.BOLD,60);                              //创建字体对象
        g2.setFont(font);                                                       //指定字体
        g2.setColor(Color.WHITE);                                               //指定颜色
```

```
        AlphaComposite alpha = AlphaComposite.SrcOver.derive(0.3f);        //获得表示透明度的 AlphaComposite 对象
        g2.setComposite(alpha);                                            //指定 AlphaComposite 对象
        g2.drawString("编程词典", -60, 180);                               //绘制文本，实现水印
        g2.dispose();
        g.dispose();                                                       //释放此图形的上下文以及它使用的所有系统资源
        ImageIO.write(image, "JPEG", response.getOutputStream());          //输出 JPEG 格式图片到页面
        response.getOutputStream().flush();                                //刷新输出流
        response.getOutputStream().close();                                //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页，在该页中添加一个<img>标签，调用 DrawWatermarkTextServlet 类生成图像并显示在页面中，关键代码如下：

```
<img alt="" src="DrawWatermarkTextServlet">
```

秘笈心法

除了使用本实例提供的方法获得表示透明度的 AlphaComposite 对象以外，还可以使用 AlphaComposite 类提供的 getInstance()方法获得 AlphaComposite 对象，如语句 AlphaComposite.getInstance(AlphaComposite.SRC_OVER) 就获得了一个规则是 AlphaComposite.SRC_OVER 的 AlphaComposite 对象（与本实例使用的字段 SrcOver 具有相同的规则），通过该对象调用 derive()方法，可以获得具有指定透明度的 AlphaComposite 对象。

13.5　图 片 特 效

实例 352	以椭圆形显示图像 光盘位置：光盘\MR\13\352	初级 实用指数：★★★☆

实例说明

在窗体上显示的图像通常都以矩形显示，本实例使用 Java 的绘图技术，实现以椭圆形显示图像。运行程序，效果如图 13.26 所示。



图 13.26　以椭圆形显示图像

关键技术

本实例的实现主要是通过图形区域的减运算，将矩形区域与椭圆形区域进行减运算，并用运算结果覆盖图像，从而实现以椭圆形显示图像的功能。

（1）使用 Area 类的构造方法封装图形对象，其构造方法的定义如下：

```
public Area(Shape s)
```

参数说明

s：是 Area 类封装的图形对象。

（2）使用 Area 类的 subtract()方法对封装的图形对象进行减运算，该方法的定义如下：

参数说明

```
public void subtract(Area rhs)
```

rhs：与当前 Area 对象进行减运算的 Area 对象。

设计过程

（1）创建以椭圆形显示图像的 Servlet 类 DrawEllipseImageServlet，然后在其 service()方法中实现绘制椭圆形图像，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
        //禁止页面缓存
        response.setHeader("Pragma", "No-cache");
        response.setHeader("Cache-Control", "No-cache");
```

```
        response.setDateHeader("Expires", 0);
        response.setContentType("image/JPEG");                         //响应格式为 JPEG 图片
        //创建一个指定长宽的图像
        int width=310, height=220;
        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        Graphics g = image.getGraphics();                              //获取用于处理图形上下文的对象
        g.setColor(new Color(221,221,221));                            //设置第 1 个矩形的 RGB 颜色
        g.fillRect(0, 0, width, height);                               //绘制实心矩形
        Graphics2D g2 = (Graphics2D)g;                                 //转换为 Graphics2D 类型
        String imgPath = getServletContext().getRealPath("/img/image.jpg");  //获取图片的路径
        BufferedImage bufferImg = ImageIO.read(new File(imgPath));     //读取图片数据，转换为 BufferImage 对象
        AffineTransformOp op = new AffineTransformOp(new AffineTransform(),null) ; //根据仿射转换和插值类型构造一个 AffineTransformOp
        g2.drawImage(bufferImg,op, 0, 0);                              //绘制图像
        Rectangle2D.Float rectangle = new Rectangle2D.Float(0, 0, width,height);  //创建矩形对象
        Ellipse2D.Float ellipse = new Ellipse2D.Float(20, 20, 260, 160);  //创建椭圆形对象
        Area area1 = new Area(rectangle);                              //创建区域矩形
        Area area2 = new Area(ellipse);                                //创建区域椭圆
        area1.subtract(area2);                                         //两个区域形状进行减运算
        g2.setColor(new Color(221,221,221));                           //设置绘图上下文的颜色为面板的背景颜色
        g2.fill(area1);                                                //绘制减运算后的区域形状
        g2.dispose();
        g.dispose();                                                   //释放此图形的上下文以及它使用的所有系统资源
        ImageIO.write(image, "JPEG", response.getOutputStream());      //输出 JPEG 格式图片到页面
        response.getOutputStream().flush();                            //刷新输出流
        response.getOutputStream().close();                            //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页，在该页中添加一个<img>标签，调用 DrawEllipseImageServlet 类生成图像并显示在页面中，关键代码如下：

```
<img alt="" src="DrawEllipseImageServlet">
```

秘笈心法

本实例实现了以椭圆形显示图像，对本实例进行扩充，可以将不同图形进行加、减、交和异或等运算，得到所需要的形状，然后用该形状覆盖原图像，就能够以该形状显示图像。

实例 353 图片百叶窗特效

光盘位置：光盘\MR\13\353

初级

实用指数：★★★☆

实例说明

本实例演示了如何利用 Java 的绘图技术，实现图片百叶窗特效。运行程序，将在网页上显示图片百叶窗效果，如图 13.27 所示。

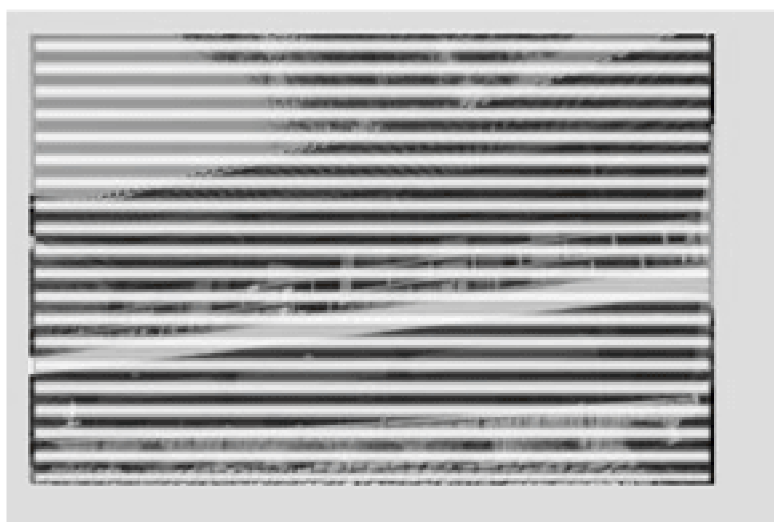


图 13.27　图片百叶窗特效

关键技术

本实例主要是在缓冲图像对象上绘制直线，并对缓冲图像对象进行模糊处理实现的，对图像进行模糊处理需要用到 Kernel 类和 ConvolveOp 类。

（1）Kernel 类定义了一个矩阵，用于描述指定的像素及其周围像素，如何影响过滤操作输出图像中像素位置的计算值，其构造方法的定义如下：

```
public Kernel(int width, int height, float[] data)
```

参数说明

❶ width：当前 Kernel 的宽度。

❷ height：当前 Kernel 的高度。

❸ data：以行优先顺序提供的 Kernel 数据。

（2）ConvolveOp 类实现从源到目标的卷积，是一种通过输入像素来计算输出像素的空间运算，方法是将核与输入像素邻域相乘。这种运算使得直接邻域可按核数学指定的方式影响输出像素，其构造方法定义如下：

```
public ConvolveOp(Kernel kernel)
```

参数说明

kernel：指定的 Kernel。

（3）ConvolveOp 类提供了一个 filter()方法，可以对缓冲图像进行过滤，实现对图像的特殊处理，如模糊、照亮边缘等，该方法的定义如下：

```
public final BufferedImage filter(BufferedImage src, BufferedImage dst)
```

参数说明

❶ src：要过滤的源 BufferedImage。

❷ dst：已过滤的 src 的目标 BufferedImage 或为 null。

❸ 返回值：对缓冲图像进行处理后的新 BufferedImage 对象。

设计过程

（1）创建用于绘制图片百叶窗特效的 Servlet 类 DrawShutterImageServlet，然后在其 service()方法中实现绘制图片的百叶窗特效，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    //禁止页面缓存
    response.setHeader("Pragma", "No-cache");
    response.setHeader("Cache-Control", "No-cache");
    response.setDateHeader("Expires", 0);
    response.setContentType("image/JPEG");                         //响应格式为 JPEG 图片
    //创建一个指定长宽的图像
    int width=340, height=230;
    BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
    Graphics g = image.getGraphics();                              //获取用于处理图形上下文的对象
    g.setColor(new Color(221,221,221));                            //设置第 1 个矩形的 RGB 颜色
    g.fillRect(0, 0, width, height);                               //绘制实心矩形
    Graphics2D g2 = (Graphics2D)g;                                 //转换为 Graphics2D 类型
    String imgPath = getServletContext().getRealPath("/img/image.jpg");  //获取图片的路径
    BufferedImage bufferImg = ImageIO.read(new File(imgPath));     //读取图片数据，转换为 BufferImage 对象
    AffineTransformOp op = new AffineTransformOp(new AffineTransform(),null); //根据仿射转换和插值类型构造一个 AffineTransformOp
    g2.drawImage(bufferImg,op, 10, 10);                            //绘制图像
    int y = 5;                                                     //直线绘制点的 y 坐标
    int space = 10;                                                //下一条直线的偏移量
    Line2D.Float line = null;
    //创建缓冲图像对象
    BufferedImage img = new BufferedImage(bufferImg.getWidth() , bufferImg.getHeight(),BufferedImage.TYPE_INT_ARGB);
    Graphics2D gs2d = (Graphics2D) img.getGraphics();              //获得缓冲图像对象的 Graphics2D 对象
    BasicStroke stroke = new BasicStroke(7);                       //创建宽度是 7 的笔画对象
    gs2d.setStroke(stroke);                                        //设置笔画对象
    gs2d.setColor(Color.WHITE);                                    //指定颜色
    while (y <=   bufferImg.getHeight()) {
        line = new Line2D.Float(0, y,bufferImg.getWidth(), y);     //创建直线对象
        gs2d.draw(line);                                           //在缓冲图像对象上绘制直线
        y = y + space;                                             //计算下一条直线的 y 坐标
    }
    for (int i = 0; i < 3; i++) {                                  //该 for 循环实现 3 次模糊
        float[] elements = new float[9];                           //定义表示像素分量的数组
        for (int j = 0; j < 9; j++) {
            elements[j] = 0.11f;                                   //为数组赋值
        }
        Kernel kernel = new Kernel(3, 3, elements);                //创建 Kernel 对象
        ConvolveOp op1 = new ConvolveOp(kernel);                   //创建 ConvolveOp 对象
        if (img == null) {
            return;
        }
        img = op1.filter(img, null);                               //过滤缓冲图像对象
```

```
        }
        g2.drawImage(img, op, 10, 10);                          //绘制缓冲图像对象
        gs2d.dispose();
        g2.dispose();
        g.dispose();                                            //释放此图形的上下文以及它使用的所有系统资源
        ImageIO.write(image, "JPEG", response.getOutputStream()); //输出 JPEG 格式图片到页面
        response.getOutputStream().flush();                     //刷新输出流
        response.getOutputStream().close();                     //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页，在该页中添加一个<img>标签，调用 DrawShutterImageServlet 类生成图像并显示在页面中，关键代码如下：

```
<img alt="" src="DrawShutterImageServlet">
```

秘笈心法

在使用 Kernel 类创建对象时，有时会出错，这是由于在使用 Kernel 类创建对象时，一定要保证数组 data 的长度，也就是 data.length 的值，必须要大于或等于 width*height，否则程序就会出错。

实例 354 图片马赛克特效

初级

光盘位置：光盘\MR\13\354

实用指数：★★★☆

实例说明

对人物或图像进行马赛克处理后可使其变得模糊，对于不想公开的内容部分可以起到保护作用。运行程序，效果如图 13.28 所示。



图 13.28　添加马赛克后的效果

关键技术

在实现本实例时，主要用到了 Graphics2D 类的 setPaint()方法和 setComposite()方法，并且应用了表示透明度的 AlphaComposite 类。有关 Graphics2D 类的 setComposite()方法以及 AlphaComposite 类的用法，请参见实例 351 的关键技术。下面主要介绍 Grahpics2D 类的 setPaint()方法。

使用 Graphics2D 类的 setPaint()方法时，需要将 GradientPaint 类创建的封装渐变颜色的对象作为 setPaint()方法的参数，实现为图形填充渐变色的操作，GradientPaint 类的构造方法的定义如下：

```
public GradientPaint(float x1, float y1, Color color1, float x2, float y2, Color color2, boolean cyclic)
```

参数说明

❶ x1：用户空间中第一个指定点的 x 坐标。

❷ y1：用户空间中第一个指定点的 y 坐标。

❸ color1：第一个指定点处的 Color 对象。

❹ x2：用户空间中第二个指定点的 x 坐标。

❺ y2：用户空间中第二个指定点的 y 坐标。

❻ color2：第二个指定点处的 Color 对象。

❼ cyclic：如果渐变模式在两种颜色之间重复循环，则该值设置为 true；否则设置为 false。

设计过程

（1）创建用于绘制图片并且为图片添加马赛克特效的 Servlet 类 DrawMosaicImageServlet，然后在其 service()方法中实现图片的马赛克特效，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
        //禁止页面缓存
        response.setHeader("Pragma", "No-cache");
```

```
        response.setHeader("Cache-Control", "No-cache");
        response.setDateHeader("Expires", 0);
        response.setContentType("image/JPEG");                          //响应格式为 JPEG 图片
        //创建一个指定长宽的图像
        int width=340, height=230;
        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        Graphics g = image.getGraphics();                               //获取用于处理图形上下文的对象
        g.setColor(new Color(221,221,221));                             //设置第 1 个矩形的 RGB 颜色
        g.fillRect(0, 0, width, height);                                //绘制实心矩形
        Graphics2D g2 = (Graphics2D)g;                                  //转换为 Graphics2D 类型
        String imgPath = getServletContext().getRealPath("/img/image.jpg");  //获取图片的路径
        BufferedImage bufferImg = ImageIO.read(new File(imgPath));      //读取图片数据，转换为 BufferImage 对象
        AffineTransformOp op = new AffineTransformOp(new AffineTransform(),null);  //根据仿射转换和插值类型构造一个 AffineTransformOp
        g2.drawImage(bufferImg,op, 10, 10);                             //绘制图像
        int x = 104;                                                    //矩形绘制点的 x 坐标
        int y = 60;                                                     //矩形绘制点的 y 坐标
        Rectangle2D.Float rect = null;
        BufferedImage img = new BufferedImage(bufferImg.getWidth() , bufferImg.getHeight(),BufferedImage.TYPE_INT_ARGB);//创建缓冲图像对象
        Graphics2D gs2d = (Graphics2D) img.getGraphics();               //获得缓冲图像对象的 Graphics2D 对象
        AlphaComposite alpha = AlphaComposite.SrcOver.derive(0.90f);    //获得表示透明度的 AlphaComposite 对象
        gs2d.setComposite(alpha);                                       //设置透明度
        GradientPaint paint = null;
        for (int i = 0; i < 4; i++) {
                for (int j = 0; j < 3; j++) {
                    paint = new GradientPaint(x, y, Color.white, x + 10,
                            y + 10, Color.gray,true);                   //创建循环渐变的 GraphientPaint 对象
                    gs2d.setPaint(paint);                               //设置渐变
                    rect = new Rectangle2D.Float(x, y, 20, 20);         //创建矩形对象
                    gs2d.fill(rect);                                    //在缓冲图像对象上绘制矩形
                    y = y + 20;                                         //计算下一个矩形的 y 坐标
                }
                y = 60;                                                 //还原 y 坐标
                x = x + 20;                                             //计算 x 坐标
        }
        for (int i = 0; i < 3; i++) {                                   //该 for 循环实现 3 次模糊
                float[] elements = new float[9];                        //定义表示像素分量的数组
                for (int j = 0; j < 9; j++) {
                    elements[j] = 0.11f;                                //为数组赋值
                }
                Kernel kernel = new Kernel(3, 3, elements);             //创建 Kernel 对象
                ConvolveOp op1 = new ConvolveOp(kernel);                //创建 ConvolveOp 对象
                if (img == null) {
                    return;
                }
                img = op1.filter(img, null);                            //过滤缓冲图像对象
        }
        g2.drawImage(img, op, 10, 10);                                  //绘制缓冲图像对象
        gs2d.dispose();
        g2.dispose();
        g.dispose();                                                    //释放此图形的上下文以及它使用的所有系统资源
        ImageIO.write(image, "JPEG", response.getOutputStream());       //输出 JPEG 格式图片到页面
        response.getOutputStream().flush();                             //刷新输出流
        response.getOutputStream().close();                             //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页，在该页中添加一个<img>标签，调用 DrawMosaicImageServlet 类生成图像并显示在页面中，关键代码如下：

```
<img alt="" src="DrawMosaicImageServlet">
```

秘笈心法

在实际应用中，如果在网页中添加的图片不想显示其中的某个指定区域，可以应用本实例中实现的方法进行设置。

实例 355 图片的模糊效果

光盘位置：光盘\MR\13\355

初级

实用指数：★★★☆

实例说明

本实例使用 Java 绘制技术实现了图片的模糊效果。运行程序，将在网页上显示图片的模糊效果，效果如图 13.29 所示。



图 13.29　图片模糊后的效果

关键技术

本实例主要是在缓冲图像对象上绘制图片，并对缓冲图像对象进行模糊处理实现的，对图像进行模糊处理需要用到 Kernel 类和 ConvolveOp 类，这两个类的具体用法请参考实例 353 的关键技术，此处不再赘述。

设计过程

（1）创建用于绘制图片模糊特效的 Servlet 类 DrawBlurImageServlet，然后在其 service()方法中实现图片的模糊特效，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    //禁止页面缓存
    response.setHeader("Pragma", "No-cache");
    response.setHeader("Cache-Control", "No-cache");
    response.setDateHeader("Expires", 0);
    response.setContentType("image/JPEG");                    //响应格式为 JPEG 图片
    //创建一个指定长宽的图像
    int width=340, height=230;
    BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
    Graphics g = image.getGraphics();                         //获取用于处理图形上下文的对象
    g.setColor(new Color(221,221,221));                       //设置第 1 个矩形的 RGB 颜色
    g.fillRect(0, 0, width, height);                          //绘制实心矩形
    Graphics2D g2 = (Graphics2D)g;                            //转换为 Graphics2D 类型
    String imgPath = getServletContext().getRealPath("/img/image.jpg");   //获取图片的路径
    BufferedImage bufferImg = ImageIO.read(new File(imgPath));            //读取图片数据，转换为 BufferImage 对象
    AffineTransformOp op = new AffineTransformOp(new AffineTransform(),null) ; //根据仿射转换和插值类型构造一个 AffineTransformOp
    BufferedImage img = new BufferedImage(bufferImg.getWidth() , bufferImg.getHeight(),BufferedImage.TYPE_INT_ARGB);//创建缓冲图像对象
    Graphics2D gs2d = (Graphics2D) img.getGraphics();         //获得缓冲图像对象的 Graphics2D 对象
    gs2d.drawImage(bufferImg, op,10,10);
    for (int i = 0; i < 3; i++) {                             //该 for 循环实现 3 次模糊
        float[] elements = new float[9];                      //定义表示像素分量的数组
        for (int j = 0; j < 9; j++) {
            elements[j] = 0.11f;                              //为数组赋值
        }
        Kernel kernel = new Kernel(3, 3, elements);           //创建 Kernel 对象
        ConvolveOp op1 = new ConvolveOp(kernel);              //创建 ConvolveOp 对象
        if (img == null) {
            return;
        }
        img = op1.filter(img, null);                          //过滤缓冲图像对象
    }
    g2.drawImage(img, op, 10, 10);                            //绘制缓冲图像对象
    gs2d.dispose();
    g2.dispose();
    g.dispose();                                              //释放此图形的上下文以及它使用的所有系统资源
    ImageIO.write(image, "JPEG", response.getOutputStream()); //输出 JPEG 格式图片到页面
    response.getOutputStream().flush();                       //刷新输出流
    response.getOutputStream().close();                       //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页，在该页中添加一个<img>标签，调用 DrawBlurImageServlet 类生成

图像并显示在页面中，关键代码如下：

```
<img alt="" src="DrawBlurImageServlet">
```

秘笈心法

本实例实现的关键是应用 ConvolveOp 类，该类提供了一个 filter()方法，用于对缓冲图像进行过滤，实现对图像的特殊处理，如模糊、照亮边缘、锐化等。

实例 356　图片的锐化效果

光盘位置：光盘\MR\13\356

初级

实用指数：★★★☆

实例说明

本实例使用 Java 绘制技术实现了图片的锐化效果。运行程序，将在网页上显示锐化效果的图片，如图 13.30 所示。



图 13.30　图片锐化后的效果

关键技术

本实例主要是在缓冲图像对象上绘制图片，并对缓冲图像对象进行锐化处理实现的，对图像进行锐化处理时，同样需要用到 Kernel 类和 ConvolveOp 类。这两个类的详细说明请参见实例 353 的关键技术。

设计过程

（1）创建用于绘制图片的锐化特效的 Servlet 类 DrawSharpenImageServlet，然后在其 service()方法中实现图片的锐化特效，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    //禁止页面缓存
    response.setHeader("Pragma", "No-cache");
    response.setHeader("Cache-Control", "No-cache");
    response.setDateHeader("Expires", 0);
    response.setContentType("image/JPEG");                              //响应格式为 JPEG 图片
    //创建一个指定长宽的图像
    int width=340, height=230;
    BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
    Graphics g = image.getGraphics();                                   //获取用于处理图形上下文的对象
    g.setColor(new Color(221,221,221));                                 //设置第 1 个矩形的 RGB 颜色
    g.fillRect(0, 0, width, height);                                    //绘制实心矩形
    Graphics2D g2 = (Graphics2D)g;                                      //转换为 Graphics2D 类型
    String imgPath = getServletContext().getRealPath("/img/image.jpg"); //获取图片的路径
    BufferedImage bufferImg = ImageIO.read(new File(imgPath));          //读取图片数据，转换为 BufferImage 对象
    AffineTransformOp op = new AffineTransformOp(new AffineTransform(),null) ; //根据仿射转换和插值类型构造一个 AffineTransformOp
    BufferedImage img = new BufferedImage(bufferImg.getWidth() , bufferImg.getHeight(),BufferedImage.TYPE_INT_ARGB);//创建缓冲图像对象
    Graphics2D gs2d = (Graphics2D) img.getGraphics();                   //获得缓冲图像对象的 Graphics2D 对象
    gs2d.drawImage(bufferImg, op,10,10);
    float[] elements = {0.0f,-1.0f,0.0f,-1.0f,5.0f,-1.0f,0.0f,-1.0f,0.0f};  //声明表示像素分量的数组
    Kernel kernel = new Kernel(3, 3, elements);                         //创建 Kernel 对象
    ConvolveOp op1 = new ConvolveOp(kernel);                            //创建 ConvolveOp 对象
    if (img == null) {
        return;
    }
    img = op1.filter(img, null);                                        //过滤缓冲图像对象
    g2.drawImage(img, op, 10, 10);                                      //绘制缓冲图像对象
    gs2d.dispose();
    g2.dispose();
    g.dispose();                                                        //释放此图形的上下文以及它使用的所有系统资源
    ImageIO.write(image, "JPEG", response.getOutputStream());           //输出 JPEG 格式图片到页面
```

```
        response.getOutputStream().flush();                          //刷新输出流
        response.getOutputStream().close();                          //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页，在该页中添加一个<img>标签，调用 DrawSharpenImageServlet 类生成图像并显示在页面中，关键代码如下：

```
<img alt="" src="DrawSharpenImageServlet">
```

秘笈心法

ConvolveOp 类可对颜色分量预乘 alpha 分量的 BufferedImage 数据进行运算。如果源 BufferedImage 有 alpha 分量，并且颜色分量没有预乘 alpha 分量，则在卷积运算前要先预乘该数据。如果 Destination 有未进行预乘的颜色分量，则在存入到 Destination 之前除以 alpha 分量（如果 alpha 为 0，则颜色分量被设置为 0）。如果 Destination 没有 alpha 分量，则在颜色分量除以 alpha 分量之后即丢弃 alpha 分量。

实例 357 图片的半透明效果

光盘位置：光盘\MR\13\357

初级

实用指数：★★★☆

实例说明

本实例使用 Java 绘制技术实现图片的半透明效果。运行程序，将在网页上显示出半透明的图片，效果如图 13.31 所示。

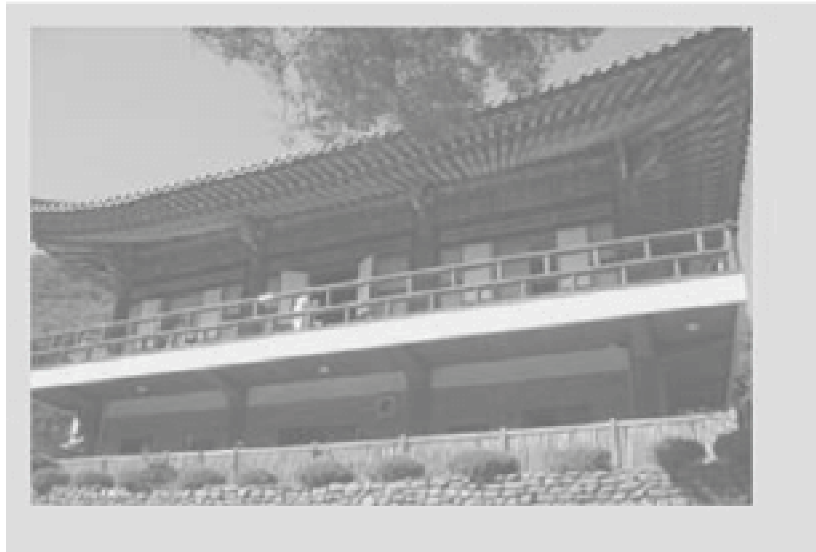


图 13.31　图片的半透明效果

关键技术

本实例主要是通过 Graphics2D 类的 setComposite()方法为绘图上下文指定表示透明度的 AlphaComposite 对象，从而实现了图片的半透明效果。具体用法请参见实例 351 的关键技术。

设计过程

（1）创建用于绘制图片的半透明特效的 Servlet 类 TranslucenceImageServlet，然后在其 service()方法中实现图片的半透明特效，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        //禁止页面缓存
        response.setHeader("Pragma", "No-cache");
        response.setHeader("Cache-Control", "No-cache");
        response.setDateHeader("Expires", 0);
        response.setContentType("image/JPEG");                       //响应格式为 JPEG 图片
        //创建一个指定长宽的图像
        int width=340, height=230;
        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        Graphics g = image.getGraphics();                            //获取用于处理图形上下文的对象
        g.setColor(new Color(221,221,221));                          //设置第 1 个矩形的 RGB 颜色
        g.fillRect(0, 0, width, height);                             //绘制实心矩形
        Graphics2D g2 = (Graphics2D)g;                               //转换为 Graphics2D 类型
        String imgPath = getServletContext().getRealPath("/img/image.jpg");  //获取图片的路径
        BufferedImage bufferImg = ImageIO.read(new File(imgPath));   //读取图片数据，转换为 BufferImage 对象
        AffineTransformOp op = new AffineTransformOp(new AffineTransform(),null) ; //根据仿射转换和插值类型构造一个 AffineTransformOp
        AlphaComposite alpha= AlphaComposite.SrcOver.derive(0.5f);   //获得表示半透明的 AlphaComposite 对象
        g2.setComposite(alpha);
        g2.drawImage(bufferImg, op, 10, 10);                         //绘制缓冲图像对象
        g2.dispose();
        g.dispose();                                                 //释放此图形的上下文以及它使用的所有系统资源
        ImageIO.write(image, "JPEG", response.getOutputStream());    //输出 JPEG 格式图片到页面
        response.getOutputStream().flush();                          //刷新输出流
```

```
        response.getOutputStream().close();                    //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页，在该页中添加一个<img>标签，调用 TranslucenceImageServlet 类生成图像并显示在页面中，关键代码如下：

```
<img alt="" src="TranslucenceImageServlet">
```

秘笈心法

本实例主要应用为绘图上下文指定表示透明度的 AlphaComposite 对象，从而实现了图片的半透明效果。

实例 358　图片的溶合效果

光盘位置：光盘\MR\13\358

初级　实用指数：★★★☆

实例说明

本实例使用 Java 的绘图技术实现了两幅图片的溶合效果。运行程序，在网页上将显示两幅图片的溶合特效，如图 13.32 所示。



图 13.32　图片溶合特效

关键技术

本实例主要是通过 Graphics2D 类的 setComposite()方法为绘图上下文设置表示透明度的 AlphaComposite 对象，从而实现了两幅图片溶合的效果。具体用法请参见实例 351 的关键技术。

设计过程

（1）创建用于绘制图片的溶合特效的 Servlet 类 PictureMixServlet，然后在其 service()方法中实现两个图片的溶合特效，关键代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
        //禁止页面缓存
        response.setHeader("Pragma", "No-cache");
        response.setHeader("Cache-Control", "No-cache");
        response.setDateHeader("Expires", 0);
        response.setContentType("image/JPEG");                 //响应格式为 JPEG 图片
        //创建一个指定长宽的图像
        int width=340, height=230;
        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        Graphics g = image.getGraphics();                      //获取用于处理图形上下文的对象
        g.setColor(new Color(221,221,221));                    //设置第 1 个矩形的 RGB 颜色
        g.fillRect(0, 0, width, height);                       //绘制实心矩形
        Graphics2D g2 = (Graphics2D)g;                         //转换为 Graphics2D 类型
        String img1Path = getServletContext().getRealPath("/img/img1.jpg");    //获取图片的路径
        String img2Path = getServletContext().getRealPath("/img/img2.jpg");    //获取图片的路径
        BufferedImage bufferImg1 = ImageIO.read(new File(img1Path));           //读取图片数据，转换为 BufferImage 对象
        BufferedImage bufferImg2 = ImageIO.read(new File(img2Path));           //读取图片数据，转换为 BufferImage 对象
        AffineTransformOp op = new AffineTransformOp(new AffineTransform(),null) ; //根据仿射转换和插值类型构造一个 AffineTransformOp
        AlphaComposite alpha = AlphaComposite.SrcOver.derive(0.5f);            //获得表示透明度的 AlphaComposite 对象
        g2.setComposite(alpha);                                //指定 AlphaComposite 对象
        g2.drawImage(bufferImg1, op,10,10);
        alpha = AlphaComposite.SrcOver.derive(0.3f);           //获得表示透明度的 AlphaComposite 对象
        g2.setComposite(alpha);                                //指定 AlphaComposite 对象
        g2.drawImage(bufferImg2, op,10,10);
        g2.dispose();
        g.dispose();                                           //释放此图形的上下文以及它使用的所有系统资源
        ImageIO.write(image, "JPEG", response.getOutputStream());  //输出 JPEG 格式图片到页面
        response.getOutputStream().flush();                    //刷新输出流
        response.getOutputStream().close();                    //关闭输出流
}
```

（2）创建用于显示图片的 index.jsp 页，在该页中添加一个<img>标签，调用 PictureMixServlet 类生成图像并显示在页面中，关键代码如下：

```
<img alt="" src="PictureMixServlet">
```

秘笈心法

本实例实现的图片溶合效果，其实并不难，主要就是将两张图片进行重叠，然后应用 AlphaComposite 设置每张图片的透明度即可达到理想效果。

实例 359 光栅图像

光盘位置：光盘\MR\13\359

初级

实用指数：★★★☆

实例说明

本实例演示如何利用 Java 的绘图技术实现光栅图像的绘制。运行程序，将在网页上显示绘制的光栅图像，效果如图 13.33 所示。

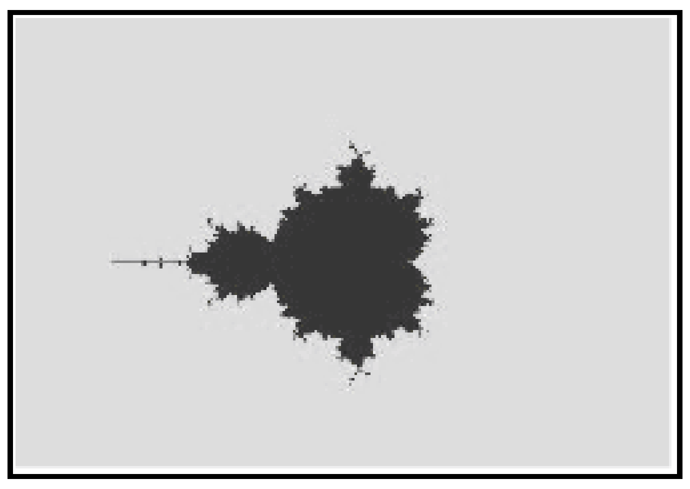


图 13.33　光栅图像

关键技术

本实例的实现主要是通过已有数字序列的数学公式，计算数字序列在指定的点（a,b）上是收敛的，还是发散的，如果是收敛的，就绘制光栅上的点；如果是发散的，则不进行光栅的绘制。

公式 x * x − y * y + a 和 2 * x * y + b 分别用于计算点（a,b）在 X 轴和在 Y 轴方向上的坐标值，如果 x 或者 y 小于等于 2 说明该点是收敛的，就绘制光栅上的点，否则说明该点是发散的，则不绘制光栅上的点。

设计过程

（1）创建用于绘制光栅图像特效的 Servlet 类 RasterImageServlet，创建 isOrNotConvergence()方法用于判断指定点是收敛的，还是发散的，关键代码如下：

```
private boolean isOrNotConvergence(double a, double b) {  //判断数字序列上的点(a,b)是收敛的，还是发散的
    double x = 0.0D;                                       //如果 x 大于 2，数字序列就是发散的
    double y = 0.0D;                                       //如果 y 大于 2，数字序列也是发散的
    int iterations = 0;                                    //循环变量
    while (x <= 2 && y <= 2 && iterations < MAX_ITERATIONS) {
        double xNew = x * x - y * y + a;                   //计算每个点的 x 值
        double yNew = 2 * x * y + b;                       //计算每个点的 y 值
        x = xNew;                                          //赋值给变量 x，用于判断是收敛还是发散
        y = yNew;                                          //赋值给变量 y，用于判断是收敛还是发散
        iterations++;                                      //调整迭代器变量的值
    }
    return x > 2 || y > 2;                                 //返回 false 表示收敛则进行绘制，为 true 表示发散则透明
}
```

（2）创建 makeRasterImage()方法，用于获得绘制有光栅的缓冲图像，关键代码如下：

```
private BufferedImage makeRasterImage(int width, int height) {
    BufferedImage image = new BufferedImage(width, height,
            BufferedImage.TYPE_INT_ARGB);                  //创建缓冲图像对象
    WritableRaster raster = image.getRaster();             //获得提供像素写入功能的 WritableRaster 对象
    ColorModel model = image.getColorModel();              //获得缓冲图像的颜色模型
    Color fractalColor = Color.RED;                        //定义表示红色的颜色对象
    int argb = fractalColor.getRGB();                      //获得表示颜色的 RGB 值
    Object colorData = model.getDataElements(argb, null);  //返回 ColorModel 中指定像素的数组表示形式
    for (int i = 0; i < width; i++) {
        for (int j = 0; j < height; j++) {
            //计算点(i,j)是否导致与点(a,b)上的像素收敛
            double a = XMIN + i * (XMAX - XMIN) / width;
```

```
                double b = YMIN + j * (YMAX - YMIN) / height;
                if (!isOrNotConvergence(a, b)) {//如果点(i,j)导致与点(a,b)上的像素收敛，escapesToInfinity()方法返回 false，则进行光栅绘制
                    raster.setDataElements((i, j, colorData);                    //为类型 TransferType 基本数组中的单个像素设置数据
                }
            }
        }
        return image;                                                           //返回绘制有光栅图像的缓冲图像对象
}
```

（3）在 service()方法中调用 makeRasterImage()方法，实现绘制光栅图像。service()方法的具体代码如下：

```
protected void service(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    //禁止页面缓存
    response.setHeader("Pragma", "No-cache");
    response.setHeader("Cache-Control", "No-cache");
    response.setDateHeader("Expires", 0);
    response.setContentType("image/JPEG");                                      //响应格式为 JPEG 图片
    //创建一个指定长宽的图像
    int width=340, height=230;
    BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
    Graphics g = image.getGraphics();                                           //获取用于处理图形上下文的对象
    g.setColor(new Color(221,221,221));                                         //设置第 1 个矩形的 RGB 颜色
    g.fillRect(0, 0, width, height);                                            //绘制实心矩形
    Graphics2D g2 = (Graphics2D)g;                                              //转换为 Graphics2D 类型
    AffineTransformOp op = new AffineTransformOp(new AffineTransform(),null) ;  //根据仿射转换和插值类型构造一个 AffineTransformOp
    g2.drawImage(this.makeRasterImage(270, 230), op,50,10);
    g2.dispose();
    g.dispose();                                                                //释放此图形的上下文以及它使用的所有系统资源
    ImageIO.write(image, "JPEG", response.getOutputStream());                   //输出 JPEG 格式图片到页面
    response.getOutputStream().flush();                                         //刷新输出流
    response.getOutputStream().close();                                         //关闭输出流
}
```

秘笈心法

本实例的实现要求有一定的数学基础，这样才能真正理解该程序，否则可以跳过该实例。

13.6　简单的验证码应用

在实际的程序开发过程中，对图像的操作非常频繁，这其中经常用到的就是验证码的功能。无论是 Web 应用程序还是桌面应用程序，应用验证码都可以有效地控制非法或暴力程序的破解，大大提高了安全性能。

实例 360	生成中文验证码 光盘位置：光盘\MR\13\360	高级 实用指数：★★

实例说明

目前，比较常见的验证都是由数字或英文字母组成的，这样的验证安全性相对较弱，为了提高验证码的安全性，可以生成中文验证码。运行本实例，在页面中将显示中文验证码，如果看不清图片中的验证文字，可以单击“看不清？换一个”超链接生成新的验证码，运行效果如图 13.34 所示。

关键技术

本实例主要应用的是生成随机数技术和汉字的编码原理，在 Java API 中提供了一个可以生成随机数的类 Random。在使用此类时需要通过实例化一个 Random 对象创建一个随机数生成器，其语法格式如下：

```
Random R = new Random();
```



图 13.34　生成中文验证码

在 Random 类中提供了获取各种类型随机数的方法，如表 13.1 所示。

说明： 以这种方式实例化对象时，Java 编译器以系统当前时间作为随机数生成器的种子，因为时间不可能相同，所以产生的随机数也将不同，但是如果运行速度太快，也会产生两次运行结果相同的随机数。

表 13.1　Random 类中的方法

方　　法	描　　述
public int nextInt	返回一个随机整数
public int nextInt(int n)	返回一个大于等于 0 小于 n 的随机整数
public long nextLong()	返回一个随机长整型值
public Boolean nextBoolean()	返回一个随机布尔型值
public float nextFloat()	返回一个随机浮点型值
public double nextDouble()	返回一个随机双精度型值
public double nextGaussian()	返回一个概率密度为高斯分布的双精度型值

例如，随机生成一个 0～20 之间的随机整数，代码如下：

```
Random random = new Random();
R3 = random.nextInt(20);
```

设计过程

（1）创建 index.jsp 页面，在页面中添加一个表单及其相关的元素属性，并在验证码的<img>标记处引用 Servlet 方法，然后<a/>超链接中使用 onClick 事件调用自定义 myReload()函数来实现更换验证码，关键代码如下：

```
<tr align="center" height="10%">
<td width="30%" align="center" valign="top">验证码：</td>
<td align="left" ><input name="checkCode" type="text" id="checkCode" size="8" maxlength="4"><br/>
        <img src="ChineseCheckCode" name="chineseCheckCode" width="120"
        height="10%" border="1" id="chineseCheckCode" />
        <a style="cursor: hand;" onClick="myReload()">  看不清?换一个</a>
</td></tr>
<tr align="center" height="10%">
<td> </td>
<td height="40" colspan="2" align="left">
<input name="submit1" type="submit"   id="submit1"value="登录" />
```

（2）创建生成验证码的 Servlet 实现类 ChineseCheckCode，此类继承自 HttpServlet，在其 service()方法中生成验证码。在生成验证码的过程中，首先设置响应头信息禁止页面缓存，并指定生成的响应是图片，关键代码如下：

```
response.setHeader("Pragma", "No-cache");
response.setHeader("Cache-Control", "No-cache");
response.setDateHeader("Expires", 0);
//指定生成的响应是图片
response.setContentType("image/jpeg");
```

（3）首先创建一个 BufferedImage 类型的图像对象，该对象创建之后只存在于缓冲区中。然后获取该图像对象的用于处理图像上下文的 Graphics 类型的对象，用于在图像上进行绘制，然后通过 Font 设置构造字体、背景等，关键代码如下：

```
int width = 166;                                          //指定验证码的宽度
int height = 45;                                          //指定验证码的高度
BufferedImage image = new BufferedImage(width, height,BufferedImage.TYPE_INT_RGB);
Graphics g = image.getGraphics();                        //获取 Graphics 类的对象
Random random = new Random();                            //实例化一个 Random 对象
Font mFont = new Font("宋体", Font.ITALIC, 26);           //通过 Font 构造字体
g.fillRect(0, 0, width, height);                         //绘制验证码背景
g.setFont(mFont);                                        //设置字体
String sRand = "";
```

（4）使用 random()方法随机生成由 4 个中文汉字组成的验证码文字，关键代码如下：

```
//输出随机的验证码文字
String ctmp = "";
int itmp = 0;
for (int i = 0; i < 4; i++) {
//生成汉字
String[] rBase = { "0", "1", "2", "3", "4", "5", "6", "7", "8","9", "a", "b", "c", "d", "e", "f" };
//生成第 1 位的区码
int r1 = random.nextInt(3) + 11;                         //生成 11～14 之间的随机数
String str_r1 = rBase[r1];
//生成第 2 位的区码
int r2;
if (r1 == 13) {
      r2 = random.nextInt(7);                            //生成 0～7 之间的随机数
} else {
      r2 = random.nextInt(16);                           //生成 0～16 之间的随机数
}
String str_r2 = rBase[r2];
//生成第 1 位的位码
int r3 = random.nextInt(6) + 10;                         //生成 10～16 之间的随机数
String str_r3 = rBase[r3];
//生成第 2 位的位码
int r4;
if (r3 == 10) {
      r4 = random.nextInt(15) + 1;                       //生成 1～16 之间的随机数
} else if (r3 == 15) {
      r4 = random.nextInt(15);                           //生成 0～15 之间的随机数
} else {
      r4 = random.nextInt(16);                           //生成 0～16 之间的随机数
}
String str_r4 = rBase[r4];
//将生成机内码转换为汉字
byte[] bytes = new byte[2];
//将生成的区码保存到字节数组的第 1 个元素中
String str_r12 = str_r1 + str_r2;
int tempLow = Integer.parseInt(str_r12, 16);
bytes[0] = (byte) tempLow;
//将生成的位码保存到字节数组的第 2 个元素中
String str_r34 = str_r3 + str_r4;
int tempHigh = Integer.parseInt(str_r34, 16);
bytes[1] = (byte) tempHigh;
ctmp = new String(bytes);                                //根据字节数组生成汉字
sRand += ctmp;
Color color = new Color(20 + random.nextInt(110));
g.setColor(color);
g.drawString(ctmp, width/6 * i+23, height/3);
```

（5）将生成的验证码保存在 Session 中，并输出生成后的验证码图片，关键代码如下：

```
//将生成的验证码保存到 Session 中
HttpSession session = request.getSession(true);
session.setAttribute("randCheckCode", sRand);
g.dispose();
ImageIO.write(image, "JPEG", response.getOutputStream());
```

秘笈心法

本实例用到了汉字的编码原理。在 1980 年为了使每一个汉字有一个全国统一的代码，我国制定了“中华人民共和国国家标准信息交换汉字编码”，标准代号为 GB2312-80，这种编码又称为国标码，在国标码的字符集中共收录了一级汉字 3755 个，二级汉字 3008 个，图形符号 682 个。

在国标 GB2312-80 中规定，所有的国标汉字及符号分配在一个 94 行 94 列的方阵中，方阵的每一行称为一个“区”，编号为 01 区～94 区，每一列称为一个“位”，编号为 01 位～94 位，方阵中的每一个汉字和符号所在的区号和位号组合在一起形成的 4 个阿拉伯数字就是它们的“区位码”。区位码的前两位是它的区号，后两位是它的位号。用区位码就可以唯一地确定一个汉字或符号，反过来说，任何一个汉字或符号也都对应着一个唯一的区位码。例如，汉字“辉”字的区位码是 2752，表明它在方阵的 27 区 52 位。

所有的汉字和符号所在的区分为以下 4 个组：

（1）01 区～15 区

图形符号区，其中 01 区～09 区为标准符号区，10 区～15 区为自定义符号区。

（2）16 区～55 区

一级常用汉字区，包括了 3755 个一级汉字。这 40 个区中的汉字是按汉语拼音排序的，同音字按笔划顺序排序。其中，55 区的 90 位～94 位未定义汉字。

（3）56 区～87 区

二级汉字区，包括了 3008 个二级汉字，按部首排序。

（4）88 区～94 区

自定义汉字区。

其中，第 10 区～15 区的自定义符号区和第 88～94 区的自定义汉字区可由用户自行定义国标码中未定义的符号和汉字。

与汉字的区位码类似的还有汉字机内码，汉字的机内码是在汉字的区位码的区码和位码上分别加上 A0H（这里的 H 表示前两位数字为十六进制数）而得到的。使用机内码表示的一个汉字占用两个字节，分别称为高位字节和低位字节，这两位字节的机内码按以下规则表示。

高位字节=区码+20H+80H（或区码+A0H）

低位字节=位码+20H+80H（或位码+A0H）

例如，汉字“啊”的区位码为 1601，区码和位码分别用十六进制表示即为 1001H，它的机内码的高位字节为 B0H，低位字节为 A1H，机内码就是 B0A1H。

实例 361	随机生成数字的验证码 光盘位置：光盘\MR\13\361	高级 实用指数：★★

实例说明

在设计登录网页时设计带验证码的用户登录模块是非常普遍的，这个登录模块是指除了正常填写用户名和密码以外，必须还要填写随机生成验证码，这样可以加强网站的安全性。运行本实例，在用户登录页面中还可以看到随机生成的数字验证码，运行结果如图 13.35 所示。

关键技术

本实例主要应用 JavaScript 脚本中的 Math 类 random()函数来生成 4 个随机数，并在 for 循环中使用，最后把随机生成的数字验证码赋值给文本框。Math 类的 random()函数的语法格式如下：

```
Math.random()
```

功能：获取 0～1 之间的伪随机数。



图 13.35　随机生成数字的验证码

若要获取 0～100 之间的数字，可以将该数字乘以 100，即 Math.random()*100；若要获取 0～50 之间的随机数，可以再与 50 进行求余运算，即(Math.random()*100)%50。

设计过程

（1）创建 index.jsp 页面，在页面中添加表单及相关元素，关键代码如下：

```
<table width="420" border="0" height="300" align="center" cellpadding="0" cellspacing="0" background="images/yzm.jpg" style="font-size:12px; color:#FFFFFF; font-weight:bold">
  <Tr height="50"><td> </td></Tr>
  <tr>
    <td height="200"><table width="98%" border="0" align="center" cellpadding="5" cellspacing="0">
        <form name="form1" method="post" >
          <tr>
            <td width="23%" height="22" align="right" class="font_white">管理员:</td>
            <td width="77%" height="22"><input name="txt_name" type="text" class="textbox" id="txt_name" size="18" maxlength="50"></td>
          </tr>
          <tr>
            <td height="22" align="right" class="font_white">密码:</td>
            <td height="22"><input name="txt_passwd" type="password" class="textbox" id="txt_passwd" size="19" maxlength="50"></td>
          </tr>
          <tr>
            <td height="22" align="right" class="font_white">验证码:</td>
           <td height="22"><input name="verifycode" id="verifycode" class="textbox" onFocus="this.select(); "onMouseOver="this.style. background =' #E1F4EE';"onMouseOut="this.style.background='#FFFFFF'" size="6" maxlength="4">
              <input name="txt_ver" type="text" id="txt_ver" size="4" readonly="ture"><input type="button" value="换一个" onclick="veri();" /></td>
          </tr>
          <tr>
          <td> </td>
            <td height="22" colspan="2" align="left"><input name="login" type="submit" id="login" value="登 录" class="button">
     
              <input type="reset" name="Submit2" value="重 置" class="button"></td>
          </tr>
        </form>
    </table></td>
  </tr>
</table>
```

（2）使用 JavaScript 脚本自定义 veri()函数，编写实现随机生成数字和字母的验证码，关键代码如下：

```
<script language="javascript">
function veri(){
var sourcenum="0123456789";
var singlenum="";
var checknum="";
var index=0;
for(i=0;i<8;i++){
    index=(Math.random()*100)%20;
    singlenum=sourcenum.substring(index,index+1);
    checknum+=singlenum;
    i++;
}
document.form1.txt_ver.value=checknum;
}
</script>
```

秘笈心法

在本实例的自定义 JavaScript 脚本 veri()函数中，先定义一个变量 sourcenum 为产生随机数的范围，然后使用 random()方法产生的随机数对 20 进行求余运算。对运算完的数值使用 subString(index,index+1)方法，是指返回 index～index+1 范围内的数值。

实例 362 生成中文、英文和数字混合的验证码

光盘位置：光盘\MR\13\362

高级

实用指数：★★

实例说明

在网站中应用验证码时，简单的验证码容易被机器人程序暴力破解。本实例实现的是中文、英文字母和数字混合的验证码，这种验证码相对来说安全性会高很多，在实际开发中这种验证码使用很常见，实例运行结果如图 13.36 所示。



图 13.36　中文、英文和数字混合的验证码

关键技术

本实例主要是在 Servlet 中的 service()方法下，实现生成汉字与英文字母验证码的过程。在生成英文字母时，可以使用 Random 类的 nextInt()方法来生成随机的 26 个英文字母的其中一个或几个。

在生成汉字时是分 4 个区码的，同样需要使用 Random 类的 nextInt()方法随机生成 4 个区码，然后组合在一起，最后应用 byte[]字节数组将生成的内码转换为汉字，再将生成的区码分别保存到字节数组的第一个元素和第二个元素中，最后根据字节数组生成汉字。

设计过程

（1）创建 index.jsp 和 deal.jsp 页面文件，在页面中添加表单及其相关的元素，关键代码如下：

```
  <tr>
    <td align="center">验证码：</td>
    <td width="18%"><input name="checkCode" type="text" id="checkCode" title="验证码区分大小写" size="8" maxlength="4"></td>
    <td width="59%" style="padding-top:4px"><img src="CheckCode" id="createCheckCode"><a href="#" onClick="myReload()"> 看不清?换一个</a></td>
  </tr>
  <tr valign="top">
    <td> </td>
    <td colspan="2"><input name="Submit" type="submit" value="登录">
       
      <input name="Submit2" type="reset" value="重置"></td>
  </tr>
</table>
```

在 deal.jsp 中验证登录的用户名、密码以及填写验证的内容是否有误，关键代码如下：

```
<%
request.setCharacterEncoding("GBK");
 String checkCode = request.getParameter("checkCode");
 if ("".equals(checkCode) || checkCode == null) {
        out.println("<script>alert('请输入验证码！ ');window.location.href='index.jsp';</script>");
 } else {
        if (!checkCode.equals(session.getAttribute("randCheckCode"))) {
                out.println("<script>alert('您输入的验证码不正确！ ');history.back(-1);</script>");
        }
 }
 if("".equals(request.getParameter("username")) || "".equals(request.getParameter("pwd"))){
        out.println("<script>alert('请输入用户名或密码！ ');window.location.href='index.jsp';</script>");
 }else{
        if(!("mr".equals(request.getParameter("username")) && "mingri".equals(request.getParameter("pwd")))){
                out.println("<script>alert('您输入的用户名或密码不正确！ ');window.location.href='index.jsp';</script>");
        }
 }
%>
```

（2）使用 JavaScript 脚本自定义函数 myReload()来重载验证码的图片，关键代码如下：

```
function myReload(){
 document.getElementById("createCheckCode").src=document.getElementById("createCheckCode").src+"?nocache="+new Date().getTime();
  }
</script>
```

（3）本实例实现需要通过 Servlet 来完成，建立在 CheckCode 类中的 service()方法下完成验证码生成的操作，首先设置响应头信息为 JPG 图片和图片的大小及字体的样式，关键代码如下：

```
response.setHeader("Pragma", "No-cache");
response.setHeader("Cache-Control", "No-cache");
response.setDateHeader("Expires", 0);
 //指定生成的响应是图片
response.setContentType("image/jpeg");
int width = 86;
int height = 22;
BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
Graphics g = image.getGraphics();
Graphics2D g2d = (Graphics2D) g;
Random random = new Random();
Font mFont = new Font("黑体", Font.BOLD, 17);
g.fillRect(0, 0, width, height);
g.setFont(mFont);
String sRand = "";
```

（4）使用 random.nextInt()方法随机生成英文 26 个字母和汉字，关键代码如下:

```
//输出随机的验证文字
String ctmp = "";
int itmp = 0;
for (int i = 0; i < 4; i++) {
        switch (random.nextInt(4)) {
                case 1:
                        itmp = random.nextInt(26) + 65;          //生成 A～Z 的字母
                        ctmp = String.valueOf((char) itmp);
                break;
                case 2:                                          //生成汉字
                        String[] rBase = { "0", "1", "2", "3", "4", "5", "6", "7", "8","9", "a", "b", "c", "d", "e", "f" };
                        //生成第 1 位的区码
                        int r1 = random.nextInt(3) + 11;         //生成 11～14 之间的随机数
                        String str_r1 = rBase[r1];
                        //生成第 2 位的区码
                        int r2;
                        if (r1 == 13) {
                                r2 = random.nextInt(7);          //生成 0～7 之间的随机数
                        } else {
                                r2 = random.nextInt(16);         //生成 0～16 之间的随机数
                        }
                        String str_r2 = rBase[r2];
                        //生成第 1 位的位码
```

```
                int r3 = random.nextInt(6) + 10;           //生成 10～16 之间的随机数
                String str_r3 = rBase[r3];
                //生成第 2 位的位码
                int r4;
                if (r3 == 10) {
                    r4 = random.nextInt(15) + 1;           //生成 1～16 之间的随机数
                } else if (r3 == 15) {
                    r4 = random.nextInt(15);               //生成 0～15 之间的随机数
                } else {
                    r4 = random.nextInt(16);               //生成 0～16 之间的随机数
                }
                String str_r4 = rBase[r4];
                //将生成的机内码转换为汉字
                byte[] bytes = new byte[2];
                //将生成的区码保存到字节数组的第 1 个元素中
                String str_r12 = str_r1 + str_r2;
                int tempLow = Integer.parseInt(str_r12, 16);
                bytes[0] = (byte) tempLow;
                //将生成的位码保存到字节数组的第 2 个元素中
                String str_r34 = str_r3 + str_r4;
                int tempHigh = Integer.parseInt(str_r34, 16);
                bytes[1] = (byte) tempHigh;
                ctmp = new String(bytes);                  //根据字节数组生成汉字
            break;
            default:
                itmp = random.nextInt(10) + 48;            //生成 0～9 的数字
                ctmp = String.valueOf((char) itmp);
            break;
        }
sRand += ctmp;
Color color = new Color(20);
g.setColor(color);
```

（5）把生成的验证码保存到 Session 中，并输出生成后的验证码图片，关键代码如下：

```
HttpSession session = request.getSession(true);
session.setAttribute("randCheckCode", sRand);
g.dispose();
```

秘笈心法

本实例中用到 Graphics 和 Graphics2D 类，Graphics2D 类对 Graphics 类进行了扩展，提供对几何形状、坐标转换、颜色管理和文本布局等更为复杂的控制。它是用于在 Java(TM) 平台上呈现二维形状、文本和图像的基础类，而 Graphics 是所有图形上下文的抽象基类，允许应用程序在组件（已经在各种设备上实现）以及闭屏图像上进行绘制，Graphics 对象封装了 Java 支持的基本呈现操作所需的状态信息。

13.7 复杂的验证码应用

简单的验证码满足不了网站的安全指标，这时可以考虑在验证码上增加背景颜色或者带多个直线、折线的干扰暴力扫描程序的攻击。下面介绍几种常见的干扰验证码的效果。

实例 363	设置验证码的字体颜色 光盘位置：光盘\MR\13\363	高级 实用指数：★★

实例说明

在开发网站中，为了达到统一的美观效果，通常设置验证码字体的颜色随机变换。运行本实例，在输入验证码处，可以看到验证码的字体颜色都是不一样的，运行效果如图 13.37 所示。

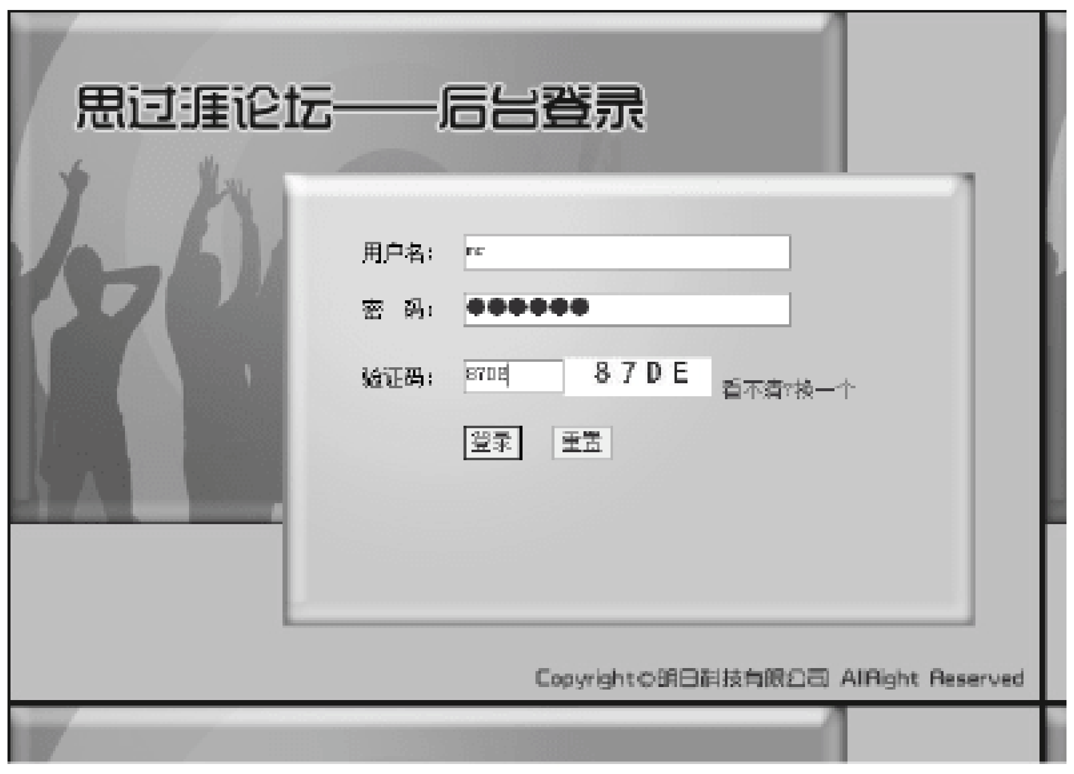


图 13.37 随机变换颜色的验证码

关键技术

本实例主要用到了 java.awt.Color 类，其构造方法的语法格式如下：

```
public Color(int r,int g,int b)
```

参数说明

❶ r：红色分量。

❷ g：绿色分量。

❸ b：蓝色分量。

Color 类是创建具有指定红色、绿色和蓝色值的不透明的 sRGB 颜色，这些值都在(0,–255)的范围内。绘制时实际使用的颜色取决于从给出的可用于给定输出设备的颜色空间中找到的最匹配颜色。alpha 值的默认值为 255，如果 r、g、b 其中的某一个值超过了 255，将会抛出 IllegalArgumentException 异常。

设计过程

（1）创建 index.jsp 和 deal.jsp 页面文件，在 index.jsp 页面中添加一个表单及其相关的元素属性，并引用 style.css 的样式，在“看不清？换一个”的<input>标记中调用 onClick 事件并加载自定义的 JavaScript 函数 myReload()，关键代码如下：

```
<tr>
  <td align="center">验证码：</td>
  <td width="18%"><input name="checkCode" type="text" id="checkCode" title="验证码区分大小写" size="8" maxlength="4"></td>
  <td width="59%" style="padding-top:4px"><img src="CheckCode" id="createCheckCode"><a href="#" onClick="myReload()"> 看不清?换一个</a></td>
</tr>
```

（2）本实例在生成验证码的过程中，需要随机生成输出内容的颜色，所以需要编写一个用于随机生成 RGB 颜色的方法 getRandColor()，返回值为 java.awt.Color 类型的颜色。getRandColor()方法的具体代码如下：

```
public Color getRandColor(int s, int e) {
        Random random = new Random();
        if (s > 255) s = 255;
        if (e > 255) e = 255;
        int r = s + random.nextInt(e - s);          //随机生成 RGB 颜色中的 r 值
        int g = s + random.nextInt(e - s);          //随机生成 RGB 颜色中的 g 值
        int b = s + random.nextInt(e - s);          //随机生成 RGB 颜色中的 b 值
        return new Color(r, g, b);
}
```

（3）在 CheckCode 类的 Servlet 中的 service()方法中调用 getRandColor()方法，用于设置验证码的字体颜色，关键代码如下：

```
g.setColor(getRandColor(100, 200));
```

由于生成验证码的 CheckCode 类的 service()方法的其他代码与前几个实例基本类似，所以此处并没有具体给出，详细代码请参见本书附赠的光盘源码。

秘笈心法

本实例主要在 CheckCode 的 Servlet 类中自定义了 getRandColor()方法随机生成 RGB 颜色，此方法有两个参

数 s 和 e 分别小于等于 255，使用 random.nextInt()方法分别随机生成 RGB 颜色 r、g、b 的值。

实例 364　具有背景颜色的验证码

光盘位置：光盘\MR\13\364

高级　实用指数：★★

实例说明

在用户登录时，通常会看到验证码的图片具有背景颜色，而且背景颜色也是随机变化的。本实例将实现这一功能，运行本实例，效果如图 13.38 所示。



图 13.38　随机变换背景颜色的验证码

关键技术

本实例的实现与实例 363 基本类似，首先获取一个随机颜色，然后再应用绘制图像上下文的 Graphics 类的 setColor()方法，在指定的验证码图像的矩形区域内填充颜色即可。

在图像上填充矩形，主要使用 Graphics 类的 fillRect()方法，语法格式如下：

```
fillRect(x,y,width,height)
```

参数说明

❶ x 和 y：表示矩形的左上角的 x 坐标和 y 坐标。

❷ width 和 height：表示矩形的宽度和高度。

设计过程

（1）创建 index.jsp 页面文件，分别在页面中添加表单及其相关元素。在自定义的 JavaScript 中检查验证码的文本框内容不允许为空，并定义 JavaScript 自定义函数 myReload()，实现在看不清验证码时进行刷新，关键代码如下：

```
<script language="javascript">
function checkform(myform){
 for(i=0;i<myform.length;i++){
        if(myform.elements[i].value==""){
                alert(myform.elements[i].title+"不能为空！");
                myform.elements[i].focus();
                return false;
        }
 }
}
 function myReload(){
        document.getElementById("createCheckCode").src=document.getElementById("createCheckCode").src+"?nocache="+new Date().getTime();
 }
</script>
   <tr>
      <td align="center">验证码：</td>
      <td width="18%"><input name="checkCode" type="text" id="checkCode" title="" size="8" maxlength="4"></td>
      <td width="59%" style="padding-top:4px"><img src="CheckCode" id="createCheckCode"><a href="#" onClick="myReload()"> 看不清?
换一个</a></td>
   </tr>
   <tr>
```

（2）在 Servlet 类中定义 getRandColor()方法，随机生成 RGB 颜色，并在 service()方法中填充指定矩形的颜色，这个矩形也就是最后生成的验证码图像，关键代码如下：

```
g.setColor(getRandColor(200, 250));//设置颜色
g.fillRect(0, 0, width, height);//填充一个矩形
```

由于生成验证码的 service()方法的其他代码与前几个实例基本类似，所以此处并没有具体给出，详细代码请参见本书附赠的光盘源码。

秘笈心法

需要注意的是，设置颜色的 setColor()方法必须在填充矩形的 fillRect()方法之前调用，否则无法实现填充颜色。这类似于画图工具，在画图之前需要设置画笔的颜色。

实例 365　随机缩放文字并将文字旋转指定角度的验证码

光盘位置：光盘\MR\13\365

高级　实用指数：★★

实例说明

在上面的几个实例中，验证码中的文字都是正常显示的，如果希望增加防破解能力，可以把汉字或字母随机缩放或将文字按照指定角度旋转，使机器人软件不容易识别。运行本实例，在加载页面后会看到在验证码图片中汉字与字母都是按随机旋转角度缩放的，如图 13.39 所示。



图 13.39　随机缩放文字并将文字旋转指定角度

关键技术

本实例主要应用 java.awt.geom.AffineTransform 类，用来实现对图像缩放或旋转操作。AffineTransform 类的构造方法的语法格式如下：

```
AffineTransform atf = new AffineTransform();
```

然后使用 AffineTransform 类提供的相关方法对图片进行缩放或旋转。其中，对图像进行缩放的方法为 scale()，对图像进行旋转的方法为 rotate()。下面对这两个方法进行详细说明。

（1）public void scale(double sx,double sy)

参数说明

❶ sx：坐标沿 X 轴方向缩放的因子。

❷ sy：坐标沿 Y 轴方向缩放的因子。

（2）public void rotate(double vecx,double vecy,double anchorx,double anchory)

参数说明

❶ vecx：旋转向量的 x 坐标。

❷ vecy：旋转向量的 y 坐标。

❸ anchorx：旋转锚点的 x 坐标。

❹ anchory：旋转锚点的 y 坐标。

连接此变换与根据旋转向量绕锚点旋转坐标的变换。所有坐标按相同的量绕指定的锚坐标旋转。旋转量使沿原 X 轴正半轴的坐标随后将与从坐标原点指向指定向量坐标的向量对齐。如果 vecx 和 vecy 都是 0.0，则不以任何方式修改此变换。此方法等效于调用：

```
rotate(Math.atan2(vecy, vecx), anchorx, anchory);
```

设计过程

（1）创建 index.jsp 页面文件，并在页面中添加一个加载验证码的<img>标签。编写自定义的 JavaScript 函数 myReload()，实现当看不清验证码时调用 Servlet 重新生成验证码，关键代码如下：

```
function myReload(){
	document.getElementById("createCheckCode").src=document.getElementById("createCheckCode").src+"?nocache="+new Date().getTime();
}
<img src="CheckCode" id="createCheckCode"><a href="#" onClick="myReload()">
```

（2）在 Servlet 的 service()方法中，应用 AffineTransform 类的相关方法实现对图片的缩放和旋转操作，关键代码如下：

```
//随机缩放文字并将文字旋转指定角度
//将文字旋转指定角度
Graphics2D g2d_word = (Graphics2D) g;
AffineTransform trans = new AffineTransform();
trans.rotate(random.nextInt(45) * 3.14 / 180, 15 * i + 8, 7);
//缩放文字
float scaleSize = random.nextFloat() +0.8f;
if (scaleSize > 1f)
	scaleSize = 1f;
trans.scale(scaleSize, scaleSize);
g2d_word.setTransform(trans);
```

秘笈心法

在 AffineTransform 类的 rotate()方法中还有几个重载方法，其中有些方法是在 JDK 1.2 中定义的，具体说明请参考 JDK API，在此就不详细说明了。

实例 366	随机生成带有干扰线的验证码 光盘位置：光盘\MR\13\366	高级 实用指数：★★

实例说明

本实例将介绍如何生成带有干扰线的验证码。运行本实例，在页面中将显示带有干扰线的验证码，用户只有输入正确的验证码后，才可完成登录，运行效果如图 13.40 所示。



图 13.40　随机生成带有干扰线的验证码

关键技术

本实例在生成验证码时，随机绘制一条具有 4 个端点的折线效果，首先定义一个端点个数的变量，然后分别定义保存 X 轴坐标的数组和 Y 轴坐标的数组，通过 for 循环为 X 轴坐标和 Y 轴坐标的数组赋值，使用 Random 类的 nextInt()方法随机生成坐标和随机位置。

应用 Graphics 类中的 drawPolyline()方法，可以绘制干扰线，该方法将使用当前颜色绘制一系列相互连接的线段，drawPolyline()方法的语法格式如下：

```
public abstract void drawPolyline(int[]xPoints,int[]yPoints,int nPoints)
```

参数说明

❶ xPoints：用于指定线段端点的 X 轴坐标。

❷ yPoints：用于指定线段端点的 Y 轴坐标。

❸ nPoints：用于指定线段端点的个数。

设计过程

（1）编写一个名称为 LineCheckCode 的 Servlet，并保存在 com.zm.servlet 包中，该类继承了 HttpServlet。主要在 service()方法中生成验证码，首先设置响应头信息禁止页面缓存，并指定生成的响应是 JPG 图片，关键代码如下：

```
response.setHeader("Pragma", "No-cache");          //禁止缓存
response.setHeader("Cache-Control", "No-cache");
response.setDateHeader("Expires", 0);
response.setContentType("image/jpeg");             //指定生成的响应是图片
```

（2）创建用于随机生成验证码的绘图类对象，并绘制一个填充矩形作为验证码的背景，关键代码如下：

```
int width = 116;                                   //指定验证码的宽度
int height = 33;                                   //指定验证码的高度
BufferedImage image = new BufferedImage(width, height,BufferedImage.TYPE_INT_RGB);
Graphics g = image.getGraphics();                  //获取 Graphics 类的对象
Random random = new Random();                      //实例化一个 Random 对象
Font mFont = new Font("宋体", Font.BOLD, 22);      //通过 Font 构造字体
g.fillRect(0, 0, width, height);                   //绘制验证码背景
g.setFont(mFont);                                  //设置字体
g.setColor(getRandColor(180, 200));                //设置颜色
```

（3）使用 Graphics 类的 drawLine()方法绘制折线，关键代码如下：

```
//画随机的线条
for (int i = 0; i < 100; i++) {
        int x = random.nextInt(width - 1);
        int y = random.nextInt(height - 1);
        int x1 = random.nextInt(3) + 1;
        int y1 = random.nextInt(6) + 1;
        g.drawLine(x, y, x + x1, y + y1);          //绘制直线
        BasicStroke bs=new BasicStroke(2f,BasicStroke.CAP_BUTT,BasicStroke.JOIN_BEVEL);
        Graphics2D g2d = (Graphics2D) g;           //通过 Graphics 类的对象创建一个 Graphics2D 类的对象
        g2d.setStroke(bs);                         //改变线条的粗细
        g.setColor(Color.GRAY);                    //设置当前颜色为预定义颜色中的灰色
        int lineNumber=4;                          //指定端点的个数
        int[] xPoints=new int[lineNumber];         //定义保存 X 轴坐标的数组
        int[] yPoints=new int[lineNumber];         //定义保存 Y 轴坐标的数组
        //通过循环为 X 轴坐标和 Y 轴坐标的数组赋值
        for(int j=0;j<lineNumber;j++){
                xPoints[j]=random.nextInt(width - 1);
                yPoints[j]=random.nextInt(height - 1);
        }                                          //绘制折线
        g.drawPolyline(xPoints, yPoints,lineNumber);
```

（4）随机生成由 4 个英文字母组成的验证码文字，并对文字进行缩放与旋转，关键代码如下：

```
String sRand = "";
//输出随机的验证文字
for (int i = 0; i < 4; i++) {
        char ctmp = (char)(random.nextInt(26) + 65);    //生成 A～Z 的字母
```

```
        sRand += ctmp;
        Color color = new Color(20 + random.nextInt(110), 20 + random
        .nextInt(110), 20 + random.nextInt(110));
        g.setColor(color);                                    //设置颜色
/** **随机缩放文字并将文字旋转指定角度* */
//将文字旋转指定角度
Graphics2D g2d_word = (Graphics2D) g;
AffineTransform trans = new AffineTransform();
trans.rotate(random.nextInt(45) * 3.14 / 180, 22 * i + 8, 7);
//缩放文字
float scaleSize = random.nextFloat() +0.8f;
if (scaleSize > 1f)
        scaleSize = 1f;
trans.scale(scaleSize, scaleSize);                            //进行缩放
g2d_word.setTransform(trans);
/** ********************** */
g.drawString(String.valueOf(ctmp), width/6 * i+23, height/2);
```

秘笈心法

本实例使用 BasicStroke 类创建一个供画笔选择线条粗细的对象，语法格式如下：

```
public BasicStroke(float width,int cap,int join)
```

参数说明

❶ width：BasicStroke 的宽度。

❷ cap：BasicStroke 端点的装饰。

❸ join：应用在路径线段交汇处的装饰。

实例 367　随机生成多条干扰线的验证码

光盘位置：光盘\MR\13\367

高级

实用指数：★★

实例说明

在增加验证码辨别难度时，还可以生成多条干扰线组合成的验证码。运行本实例，将在页面中显示出带有多条干扰线的验证码，只有用户输入正确的验证码后才可完成登录。如果随机生成的验证码不容易识别，可以单击“换一个”超链接生成新的验证码，运行效果如图 13.41 所示。



图 13.41　随机生成多条直线的干扰线

关键技术

绘制直线，主要应用的是 Graphics 类的 drawLine()方法，该方法用于使用当前颜色绘制一条直线。drawLine()方法的语法格式如下：

```
public abstract void drawLine(int x1,int y1,int x2,int y2)
```

参数说明

❶ x1：用于指定直线起点的 X 轴坐标。

❷ y1：用于指定直线起点的 Y 轴坐标。

❸ x2：用于指定直线起点的 X 轴坐标。

❹ y2：用于指定直线起点的 Y 轴坐标。

设计过程

（1）创建 index.jsp 页面，在页面中添加表单及其相关元素，关键代码如下：

```
<tr>
    <td align="center">验证码：</td>
```

```
    <td style="padding-top:4px"><input name="checkCode" type="text" width="33" height="116" id="checkCode" title="验证码区分大小写"
size="8" maxlength="4">
<img src="LineCheckCode" id="createCheckCode" >
    <a href="#"   onclick="myReload();">换一个</a></td>
</tr>
```

（2）编写一个名为 LineCheckCode 的 Servlet 类，主要是通过 service()方法来实现生成验证码。在 service()方法中，通过循环在验证码图像上随机的两个坐标点之间画 100 条干扰线，关键代码如下：

```
for (int i = 0; i < 100; i++) {
        int x = random.nextInt(width - 1);
        int y = random.nextInt(height - 1);
        int x1 = random.nextInt(3) + 1;
        int y1 = random.nextInt(6) + 1;
        g.drawLine(x, y, x + x1, y + y1);      //绘制直线的方法
}
```

秘笈心法

由于在应用 drawLine()方法画直线时，每条线的两个端点的坐标是随机产生的，所以最后显示在验证码图片上的线段也是随机的，这样就实现了干扰线的效果。如果希望画更多的干扰线，只要修改循环次数即可。

实例 368	随机生成关键字验证码 光盘位置：光盘\MR\13\368	高级 实用指数：★★

实例说明

关键字验证码是指在验证码显示框中，不再只是显示验证码文字，而是由具有指定意义的一组字符串组成，对于验证码关键字采用特殊颜色进行标识，用户只要按顺序输入指定颜色的字符，即可正确输入验证码。这种验证码的安全性比较高，对于一些涉及财务的网站比较实用，实例运行结果如图 13.42 所示。

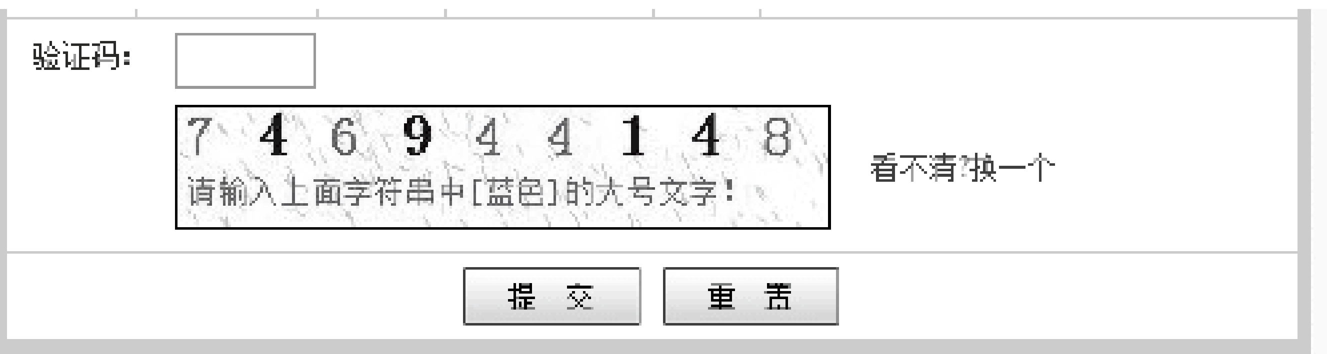


图 13.42　生成关键字验证码

关键技术

实现本实例时，首先使用 Graphics 类的 drawLine()方法绘制了 150 条浅灰色的随机直线显示在验证码中，然后在获取 4 个要突出显示的验证码关键字时，需要记录关键字的位置变量和获取字符串的长度，并使用 while()循环语句来显示关键字在验证码中的位置。

注意：为了保证生成的验证码关键字为 4 个，此处不能使用 for 循环语句，因为在随机获取要突出显示的文字位置时，可能会产生相同的值，这样最后生成的验证码关键字就不是 4 个了。

设计过程

（1）编写一个名称为 KeywordCheckcode 的 Servlet 类，在 service()方法中设置响应头信息并指定响应是 JPG 图片，然后创建用于生成验证码的绘图类对象，并绘制一个填色矩形作为验证码的背景，关键代码如下：

```
response.setHeader("Pragma", "No-cache");
response.setHeader("Cache-Control", "No-cache");
response.setDateHeader("Expires", 0);
//指定生成的响应是图片
response.setContentType("image/jpeg");
int width = 270;                              //设置验证码的宽度
int height = 50;                              //设置验证码的高度
```

```
BufferedImage image = new BufferedImage(width, height,BufferedImage.TYPE_INT_RGB);
Graphics g = image.getGraphics();                    //获取 Graphics 类的对象
Random random = new Random();                        //实例化一个 Random 对象
g.fillRect(0, 0, width, height);                     //绘制验证码背景
g.setColor(new Color(0xCCCCCC));                     //设置颜色
```

（2）随机绘制 150 条浅灰色的随机直线，关键代码如下：

```
//画随机的线条
for (int i = 0; i < 150; i++) {
        int x = random.nextInt(width - 1);
        int y = random.nextInt(height - 1);
        int x1 = random.nextInt(3) + 1;
        int y1 = random.nextInt(6) + 1;
        g.drawLine(x, y, x + x1, y + y1);            //绘制直线
}
```

（3）获取生成验证码的字符串和获取 4 个要突出显示的验证关键字在验证码字符串的位置，组成由逗号（,）分隔的字符串，关键代码如下：

```
String str=request.getParameter("number");           //获取生成验证码的字符串
String sRand = "";
String randomPosition=",";                           //记录关键字位置的变量
int strLength=str.length();                          //获取字符串的长度
int r=0;
int i=0;
while(i<4){                                          //注意此处不能使用 for 循环语句
        r=random.nextInt(strLength-1);
        if(randomPosition.indexOf(","+r+",")==-1){
                randomPosition+=String.valueOf(r)+",";
                i++;
        }
}
```

（4）使用 Color 类编写生成随机的关键字颜色，并通过 for 循环输出全部验证码文字，普通文字为固定的深色，验证码关键颜色为指定的随机颜色，关键代码如下：

```
String colorValue="";
Color color=Color.RED;
switch (random.nextInt(3)){
      case 0:
            colorValue="红色";
            color=Color.RED;
            break;
      case 1:
            colorValue="黑色";
            color=Color.BLACK;
            break;
      case 2:
            colorValue="蓝色";
            color=Color.BLUE;
            break;
}
for (int j = 0; j < strLength; j++) {
        if(randomPosition.indexOf(","+String.valueOf(j)+",")==-1){
                g.setFont(new Font("宋体", Font.BOLD, 22));
                g.setColor(new Color(0x009900));     //设置颜色为深绿色
        }else{
                g.setFont(new Font("宋体", Font.BOLD, 24));
                g.setColor(color);
                sRand+=str.substring(j,j+1);
        }
        g.drawString(str.substring(j,j+1), width/(strLength) * j+3, 20);

}
```

（5）输出提示性文字，说明关键字的颜色，关键代码如下：

```
//输出提示性文字
g.setFont(new Font("宋体", Font.PLAIN, 13));
g.setColor(new Color(0x009900));
g.drawString("请输入上面字符串中["+colorValue+"]的大号文字！", 3, 40);
```

（6）最后创建 index.jsp 首页文件，并在合适位置添加表单及其相关表单元素，并在页面中添加一个调用 myReload()函数的超链接来实现重新生成验证码的功能，关键代码如下：

```
<tr>
    <td width="13%" height="64" align="center"> </td>
    <td width="54%" align="left">
    <img src="KeywordCheckCode?number=${number}" name="keywordCheckCode" width="270" height="50" border="1" id="keywordCheckCode"/>
</td>
    <td width="33%" align="left"><a   style="cursor:hand;"onClick="myReload()">看不清?换一个</a></td>
</tr>
```

秘笈心法

本实例主要应用其 indexOf()方法获取字符串的索引位置，indexOf()方法的语法格式如下：

```
Public int indexOf(Sting str)
```

其中，参数 str 为任意字符串，用于指定要搜索的子字符串。

说明：indexOf()方法用于返回指定子字符串在此字符串中第一次出现的索引位置，如果未发现该字符串，则返回-1。

实例 369　利用 Ajax 实现无刷新的彩色验证码　高级

光盘位置：光盘\MR\13\369　　实用指数：★★

实例说明

相对于英文、中文、数字混合的彩色验证码来说，利用 Ajax 实现的验证码更加安全。这种验证码不是在页面加载后生成，而是当用户单击验证码输入框时才生成并显示，这样就能有效地防止恶意用户采用注册机类的暴力程序对网站进行攻击或灌水。实例运行结果如图 13.43 所示，在该页面中输入留言人和留言内容后，单击“验证码”文本框，将在其上方生成并显示验证码，用户输入验证码并将光标移出文本框时，系统将自动验证输入的验证码是否正确，并给出提示信息。



图 13.43　Ajax 实现无刷新的验证码

关键技术

实现本实例，首先在 Servlet 中随机生成一个验证码图片，因为验证码图片开始时未显示在页面中，它是由鼠标的光标移动到验证码文本框内才会在上方显示，所以在页面中插入验证码显示框时，需要添加 style 属性的 display 属性并设置为 none，用于指定该<div>标记默认为不显示。

关于 Ajax 技术读者可以参见本书第 10 章的说明，在此不再详细介绍。

设计过程

（1）编写一个名称为 CheckCode.java 的 Servlet 类，在 service()方法中实现生成验证码。首先在 service()方法中绘制一条干扰折线，颜色设置为深灰色且位置随机产生，关键代码如下：

```
//创建一个供画笔选择线条粗细的对象
BasicStroke bs=new BasicStroke(2f,BasicStroke.CAP_BUTT,BasicStroke.JOIN_BEVEL);
g2d.setStroke(bs);                          //改变线条的粗细
```

```
g.setColor(Color.DARK_GRAY);                    //设置当前颜色为预定义颜色中的深灰色
int[] xPoints=new int[3];
int[] yPoints=new int[3];
for(int j=0;j<3;j++){
        xPoints[j]=random.nextInt(width - 1);
        yPoints[j]=random.nextInt(height - 1);
}
```

（2）随机生成 4 个英文和数字混合的验证码文字，并对文字进行随机缩放并旋转，关键代码如下：

```
g.setFont(mFont);
String sRand="";
int itmp=0;
for(int i=0;i<4;i++){
        if(random.nextInt(2)==1){
                itmp=random.nextInt(26)+65;             //生成 A～Z 的字母
        }else{
                itmp=random.nextInt(10)+48;             //生成 0～9 的数字
        }
        char ctmp=(char)itmp;
        sRand+=String.valueOf(ctmp);
        Color color=new Color(20+random.nextInt(110),20+random.nextInt(110),20+random.nextInt(110));
        g.setColor(color);
        /****随机缩放文字并将文字旋转指定角度**/
        //将文字旋转指定角度
        Graphics2D g2d_word=(Graphics2D)g;
        AffineTransform trans=new AffineTransform();
        trans.rotate(random.nextInt(45)*3.14/180,15*i+10,7);
                                                        //缩放文字
        float scaleSize=random.nextFloat()+0.8f;
        if(scaleSize>1.1f) scaleSize=1f;
        trans.scale(scaleSize, scaleSize);
        g2d_word.setTransform(trans);
        g.drawString(String.valueOf(ctmp),30*i+40,16);
}
```

（3）创建 index.jsp 首页文件，在页面中插入验证码显示框，在适当位置添加一个<div>标记，将其 style 属性的子属性 position 设置为 absolute，用于确定要显示的验证码的位置，关键代码如下：

```
<div style="position:absolute">
        <div id="showCheckCode" style="display:none; padding:3px" align="center" >
                <img   src="images/checkCodePicture.jpg" id="createCheckCode" width="200" height="60">
                <a href="#" style="color:#EEEEEE" onClick="getCheckCode1(showCheckCode,checkCode)">看不清?换一个</a>
        </div>
         <input name="checkCode" type="text" id="checkCode" size="6" value="" title="单击验证码输入框，获取验证码"
 onClick="getCheckCodeFun(showCheckCode,checkCode);"    onBlur="checkCheckCode(this.value)">
</div>
```

（4）通过 Ajax 调用生成彩色验证码的 Servlet 生成验证码，创建一个单独的 JavaScript 文件，名称为 AjaxRequest.js，并在该文件中编写重构 Ajax 所需的代码，关键代码如下：

```
var net=new Object();
//编写构造函数
net.AjaxRequest=function(url,onload,onerror,method,params){
   this.req=null;
   this.onload=onload;
   this.onerror=(onerror) ? onerror : this.defaultError;
   this.loadDate(url,method,params);
}
//编写用于初始化 XMLHttpRequest 对象并指定处理函数，最后发送 HTTP 请求的方法
net.AjaxRequest.prototype.loadDate=function(url,method,params){
   if (!method){
      method="GET";
   }
   if (window.XMLHttpRequest){
      this.req=new XMLHttpRequest();
   } else if (window.ActiveXObject){
      this.req=new ActiveXObject("Microsoft.XMLHTTP");
   }
   if (this.req){
      try{
         var loader=this;
```

```
            this.req.onreadystatechange=function(){
                net.AjaxRequest.onReadyState.call(loader);
            }
            this.req.open(method,url,true);
        if(method=="POST"){
            this.req.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
        }
            this.req.send(params);
        }catch (err){
            this.onerror.call(this);
        }
    }
}
//重构回调函数
net.AjaxRequest.onReadyState=function(){
    var req=this.req;
    var ready=req.readyState;
    if (ready==4){
        if (req.status==200 ){
            this.onload.call(this);
        }else{
            this.onerror.call(this);
        }
    }
}
net.AjaxRequest.prototype.defaultError=function(){
    alert("错误数据\n\n 回调状态:"+this.req.readyState+"\n 状态: "+this.req.status);
}
```

（5）编写请求处理页面文件 getCheckCode.jsp，在该文件中将调用生成彩色验证码的 Servlet 生成验证码，getCheckCode.jsp 文件的关键代码如下：

```
<%@ page contentType="text/html; charset=GBK" language="java" import="java.util.Random"%>
<%Random random = new Random();%>
<img src="<% out.println("CheckCode");%>" id="createCheckCode" width="200" height="60">
<a href="#" style="color:#EEEEEE" onclick="getCheckCode1(showCheckCode,checkCode)">看不清?换一个</a>
```

（6）在 index.jsp 页面中编写实例化 Ajax 对象的方法和回调函数，关键代码如下：

```
function getCheckCode1(showCheckCode,checkCode){
        var loader1=new net.AjaxRequest("getCheckCode.jsp?nocache="+new Date().getTime(),deal_getCheckCode,onerror,"GET");
        showCheckCode.style.display='';
        checkCode.focus();
}
function deal_getCheckCode(){
        document.getElementById("showCheckCode").innerHTML=this.req.responseText;
}
```

（7）在 index.jsp 页面中编写自定义的 JavaScript 函数，用于根据实际情况生成并显示验证码，关键代码如下：

```
//生成并显示验证码
function getCheckCodeFun(showCheckCode,checkCode){
 showCheckCode.style.display='';
 if(document.getElementById("resultMessage").innerHTML=="温馨提示：单击验证码输入框，获取验证码！验证码区分大小写"){
        getCheckCode1(showCheckCode,checkCode);  //生成验证码
 }
 checkCode.focus();
}
```

在页面的合适位置添加验证码输入框，在该文本框的 onClick 事件中调用自定义 JavaScript 函数 getCheckCodeFun()，关键代码如下：

```
<input name="checkCode" type="text" id="checkCode" size="6" value="" title="单击验证码输入框，获取验证码"
 onClick="getCheckCodeFun(showCheckCode,checkCode);" >
```

（8）由于采用了 Ajax 技术，所以可以很方便地实现无刷新检测验证是否正确，首先编写检测验证码是否正确时，应用实例化 Ajax 对象的方法和回调函数，关键代码如下：

```
function deal_checkCheckCode(){
 var h=this.req.responseText;
 h=h.replace(/\s/g,"");                          //去除字符串中的 Unicode 空白符
 if(h==1){
```

```
        document.getElementById("resultMessage").removeChild(document.getElementById("resultMessage").childNodes[0]);
        document.getElementById("resultMessage").appendChild(document.createTextNode(" "));
        document.getElementById("messageImg").src="images/dui2.gif";
        document.getElementById("resultMessage").removeChild(document.getElementById("resultMessage").childNodes[0]);
        document.getElementById("resultMessage").appendChild(document.createTextNode("恭喜您，验证码正确！"));
        document.getElementById("btn_Submit").disabled=false;
}else{
        document.getElementById("messageImg").src="images/cuo2.gif";
        document.getElementById("resultMessage").removeChild(document.getElementById("resultMessage").childNodes[0]);
        document.getElementById("resultMessage").appendChild(document.createTextNode("您输入的验证码不正确！"));
}
}
</script>
```

（9）创建 checkCheckCode.jsp 页，验证输入的验证码是否正确，在该文件中将用户输入的验证码与 Session 中的验证码进行比较，如果相等则输出 1，表示验证码正确，否则输出 0，表示验证码不正确，关键代码如下：

```
<%@ page contentType="text/html; charset=GBK" language="java" %>
<%
String inCheckCode=request.getParameter("inCheckCode");
 if(session.getAttribute("randCheckCode").equals(inCheckCode)){
        out.println("1");
}else{
        out.println("0");
}
%>
```

秘笈心法

本实例应用 Ajax 实现无刷新的彩色验证码后，在 IE 浏览器中测试，运行一切正常，但是把该实例放在 Firefox 浏览器中，单击“看不清？换一个”超链接后，出现了验证码不改变的情况。这是因为在 Firefox 浏览器中缓存了验证码图片，这时可以在请求处理页面 getCheckCode.jsp 中，将图片的 src 属性通过 JSP 代码动态传递一个参数，参数值为随机数即可。

修改后的 getCheckCode.jsp 文件的代码如下：

```
<%@ page contentType="text/html; charset=GBK" language="java"    import="java.util.Random"%>
<%Random random = new Random();%>
<img src="<% out.println("CheckCode?rand="+random.nextInt(10000));%>" id="createCheckCode" width="200" height="60">
<a href="#" style="color:#EEEEEE" onclick="getCheckCode1(showCheckCode,checkCode)">看不清?换一个</a>
```

实例 370	生成带雪花的验证码 光盘位置：光盘\MR\13\370	高级 实用指数：★★

实例说明

为了提高网站的安全性，生成带有背景和雪花的验证码可以大大提高验证码的识别难度，从而有效阻止不法分子通过注册机类的程序对网站进行攻击。本实例将介绍如何实现生成带有雪花的验证码。运行本实例，在页面中将显示雪花的验证码，用户只有输入正确的验证码后，才可完成注册，运行结果如图 13.44 所示。



图 13.44　带雪花的验证码

关键技术

实现本实例，首先在 Servlet 中定义一个用于存储雪花数量的变量 snowNumber，然后编写一个随机变换颜

色的方法，用来重新设置雪花的颜色，使用 for()语句来循环 snowNumber 变量，然后创建 Graphics2D 类的对象和 AffineTransform 类的对象，用于对生成的雪花进行旋转，代码如下：

```
trans.rotate(random.nextInt(45) * 3.14 / 180, random.nextInt(width / 2), 7)
```

另外，使用“*”号来代替雪花图案，只要随机设置“*”号的宽度和高度即可。

设计过程

（1）编写一个名称为 SnowCheckCode 的 Servlet 类，在 service()方法中生成验证码。在 SnowCheckCode 类中定义一个全局变量，用于指定默认的雪花个数，关键代码如下：

```
protected int snowNumber=20;                    //指定雪花个数
```

（2）创建用于生成验证码的绘图类对象和指定个数的雪花，颜色为白色，位置随机产生，并且对雪花进行随机缩放和旋转，关键代码如下：

```
int width = 166;                                //验证码图片的宽度
int height = 45;                                //验证码图片的高度
BufferedImage image = new BufferedImage(width, height,BufferedImage.TYPE_INT_RGB);
Graphics g = image.getGraphics();               //获取 Graphics 类的对象
/*********************************************************/
Random random = new Random();                   //实例化一个 Random 对象
g.setColor(getRandColor(200, 250));
g.fillRect(0, 0, width, height);
g.setFont(new Font("宋体", Font.PLAIN, 16));     //设置字体
g.setColor(new Color(255, 255, 255));           //设置雪花的颜色
//画随机的雪花
for (int i = 0; i < snowNumber; i++) {
    /** **随机缩放雪花并将其旋转指定角度* */
    //将雪花旋转指定角度
    Graphics2D g2d = (Graphics2D) g;            //通过 Graphics 类的对象创建一个 Graphics2D 类的对象
    AffineTransform trans = new AffineTransform();   //创建 AffineTransform 类的对象
    trans.rotate(random.nextInt(45) * 3.14 / 180, random.nextInt(width / 2), 7);//进行旋转
    float scaleSize = random.nextFloat() + 0.5f;
    trans.scale(scaleSize, scaleSize);          //进行缩放
    g2d.setTransform(trans);
    /** ******************** */
    g.drawString("*", random.nextInt(width), random.nextInt(height / 2));
}
```

（3）随机生成 4 个英文和数字混合的验证码文字，并对文字进行随机缩放和旋转，关键代码如下：

```
String sRand = "";
Font mFont = new Font("宋体", Font.ITALIC, 26);   //通过 Font 构造字体
g.setFont(mFont);                               //设置字体
g.setColor(new Color(0, 0, 0));                 //设置颜色为黑色
//输出随机的验证码文字
int itmp = 0;
for (int i = 0; i < 4; i++) {
    if (random.nextInt(2) == 1) {
        itmp = random.nextInt(26) + 65;         //生成 A～Z 的字母
    } else {
        itmp = random.nextInt(10) + 48;         //生成 0～9 的数字
    }
    char ctmp = (char) itmp;
    sRand += String.valueOf(ctmp);
    /****随机缩放文字并将文字旋转指定角度**/
    //将文字旋转指定角度
    Graphics2D g2d_word = (Graphics2D) g;
    AffineTransform trans = new AffineTransform();
    //缩放文字
    float scaleSize = random.nextFloat() + 0.5f;
    if (scaleSize < 0.8 || scaleSize > 1.1f) {
        scaleSize = 1f;
    }
    trans.scale(scaleSize, scaleSize);
    g2d_word.setTransform(trans);
```

```
    /***********************/
    g.drawString(String.valueOf(ctmp), width / 6 * i + 23, random.nextInt(height / 2) + 20);
}
```

秘笈心法

对雪花的 width 和 height 随机变换大小用到了 drawString()方法，其语法格式如下：

```
public abstact void drawSting(String str,int x,int y)
```

参数说明

❶ str：是需要绘制的 String 字符串。

❷ x：x 坐标。

❸ y：y 坐标。

实例 371 生成带背景的验证码

光盘位置：光盘\MR\13\371

高级

实用指数：★★

实例说明

为了加强验证码的识别难度，还可以生成带有背景的验证码。本实例将介绍如何生成带有背景图片的验证码。运行本实例，当页面被加载后，在验证码处可以看到有指定图片作为验证码的背景，只有用户输入正确的验证码后，才可完成注册，运行结果如图 13.45 所示。

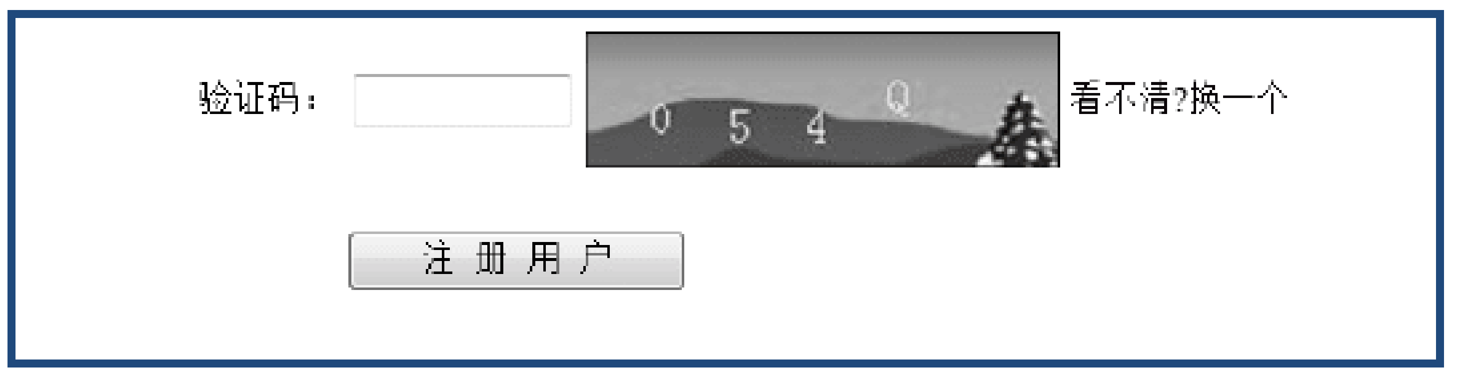


图 13.45 背景图片的验证码

关键技术

在获取验证码背景时，首先需要创建作为背景图片的文件对象（这可以通过 java.io 包中的 File 类的构造方法实现），然后再构造一个 Image 对象，最后通过 Graphics 类的 drawImage()方法输出背景图片。

（1）构造 Image 对象

Image 对象可以通过 ImageIO 类中的 read()方法获取。ImageIO 类是 Java 管理图像输入/输出的类，它位于 javax.imageio.ImageIO 包中，本实例涉及了该类的 read()方法，使用它读取指定的图像文件，该方法是静态方法，所以不用创建 ImageIO 类的实例对象，直接就可以调用，语法格式如下：

```
read(File imgFile)
```

参数说明

imgFile：一个文件对象，该文件对象指向一个图像文件。

例如，本实例中获取验证码背景图片所对应的 Image 对象的代码如下：

```
Image src = ImageIO.read(bgImgFile);    //构造 Image 对象
```

（2）通过 Graphics 类的 drawImage()方法绘制背景图片

使用 Java 的绘图类 Graphics，不仅可以绘制图形和文本，还可以使用其 drawImage()方法将图片资源显示到绘图上下文中，而且可以实现各种特效处理。Graphics 类的 drawImage()方法的语法格式如下：

```
drawImage(Image img,int x,int y,ImageObserver observer)
```

参数说明

❶ img：要显示的图片对象。

❷ x：水平位置。

❸ y：垂直位置。

❹ observer：要通知的图像观察者。

例如，本实例中通过 Graphics 类的 drawImage()方法绘制背景图片的代码如下：

```
g.drawImage(src, 0, 0, width, height, null);                    //绘制背景图片
```

设计过程

（1）编写一个名称为 SnowCheckCode 的 Servlet 类，主要通过 service()方法生成验证码。首先设置响应头信息并指定生成的响应是 JPG 图片和获取验证码的背景图片，并绘制图片作为验证码的背景，关键代码如下：

```
response.setHeader("Pragma", "No-cache");
response.setHeader("Cache-Control", "No-cache");
response.setDateHeader("Expires", 0);
//指定生成的响应是图片
response.setContentType("image/jpeg");
int width = 166;                                     //验证码图片的宽度
int height = 45;                                     //验证码图片的高度
BufferedImage image = new BufferedImage(width, height,BufferedImage.TYPE_INT_RGB);
Graphics g = image.getGraphics();                    //获取 Graphics 类的对象
//获取背景图片
File bgImgFile = new File(request.getRealPath("images/checkCode_bg.jpg"));
Image src = ImageIO.read(bgImgFile);                 //构造 Image 对象
g.drawImage(src, 0, 0, width, height, null);         //绘制背景图片
```

（2）随机生成 4 个英文和数字混合的验证码文字，并对文字进行随机缩放并旋转，关键代码如下：

```
String sRand = "";
Font mFont = new Font("宋体", Font.ITALIC, 26);        //通过 Font 构造字体
g.setFont(mFont);                                      //设置字体
g.setColor(new Color(0, 0, 0));                        //设置颜色为黑色
//输出随机的验证码文字
int itmp = 0;
for (int i = 0; i < 4; i++) {
    if (random.nextInt(2) == 1) {
            itmp = random.nextInt(26) + 65;            //生成 A～Z 的字母
    } else {
            itmp = random.nextInt(10) + 48;            //生成 0～9 的数字
    }
    char ctmp = (char) itmp;
    sRand += String.valueOf(ctmp);
    /****随机缩放文字并将文字旋转指定角度**/
    //将文字旋转指定角度
    Graphics2D g2d_word = (Graphics2D) g;
    AffineTransform trans = new AffineTransform();
    //缩放文字
    float scaleSize = random.nextFloat() + 0.5f;
    if (scaleSize < 0.8 || scaleSize > 1.1f) {
          scaleSize = 1f;
    }
    trans.scale(scaleSize, scaleSize);
    g2d_word.setTransform(trans);
    g.drawString(String.valueOf(ctmp), width / 6 * i + 23, random.nextInt(height / 2) + 20);
}
```

（3）将生成的验证码保存到 Session 中，并输出生成后的验证码，关键代码如下：

```
HttpSession session = request.getSession(true);
session.setAttribute("randCheckCode", sRand);
g.dispose();
ImageIO.write(image, "JPEG", response.getOutputStream());
```

秘笈心法

本实例应用了 Graphics 类的 drawImage()方法绘制背景图片，该方法还有其他几个重载方法，这些方法的用法可以参考 JDK API。

13.8　生成条形码

实例 372　利用组件生成条形码　　**高级**

光盘位置：光盘\MR\13\372　　实用指数：★★

实例说明

相信大家都在超市或者书店见过一种扫描设备，它就是条形码，是一种用于区分产品的一组编号，以条形码呈现给用户，如图书上的 ISBN 编号、各种电子产品和保修凭证上的条形码等。本实例在 JSP 页面实现了条形码的在线生成功能，如图 13.46 所示，用户在页面中可以输入要生成条形码的产品编号，也可以包含英文字母，单击“生成条形码”按钮，将生成各种编码格式的条形码图片，如图 13.47 所示。

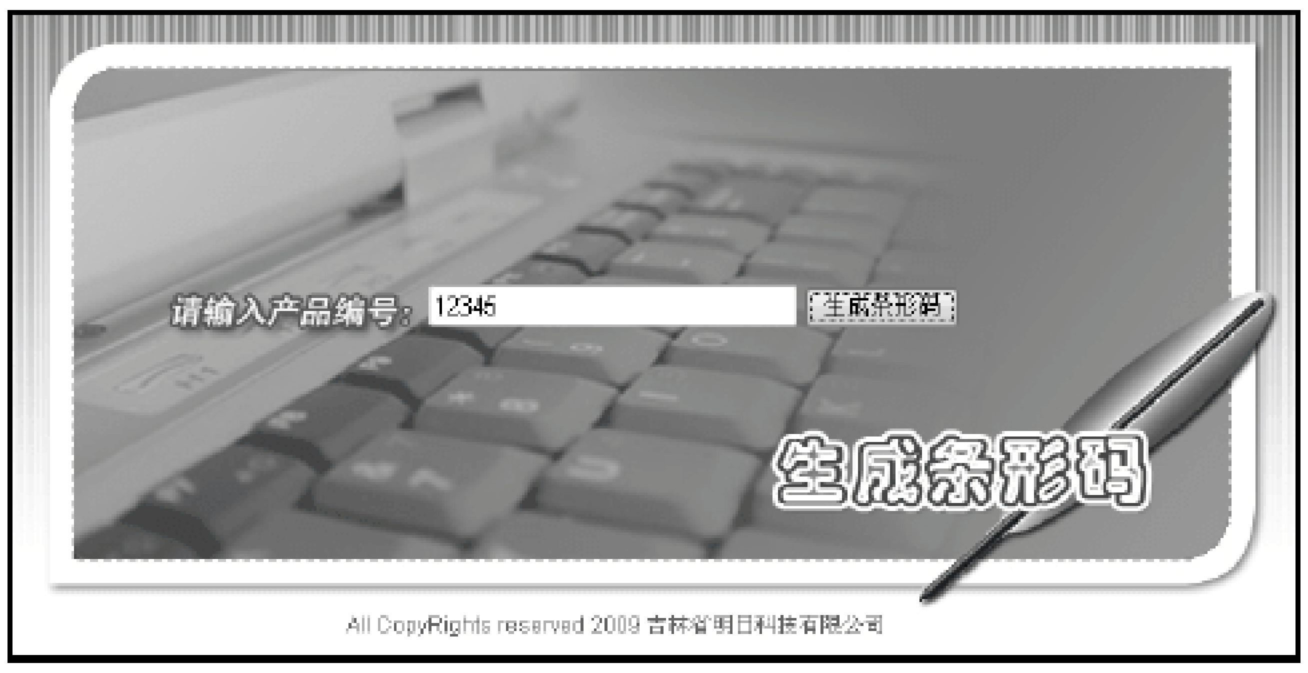


图 13.46　输入要生成的条形码



图 13.47　生成的条形码

关键技术

本实例主要使用了开源 JbarcodeBean 组件，该组件主要用于生成条形码。其中主要应用该组件包中的 JBarcodeBean 类中的相关方法生成条形码。JBarcodeBean 类中用于设置条形码样式的常用方法如表 13.2 所示。

表 13.2　JBarcodeBean 类的常用方法

方　法	描　述
setCodeType(BarcodeStrategy codeType)	设置生成的条形码类型
setCode(String code)	设置要生成的条形码
setBarcodeHeight(int height)	设置条形码的高度
setFont(Font font)	设置字体
setForeground(Color c)	设置前景色
setBarcodeBackground(Color c)	设置条形码的背景色

设计过程

（1）编写 index.jsp 主页面，在其中定义接收产品编号的文本框和执行条形码生成的提交按钮以及相关元素，关键代码如下：

```
<form method="get" action="result.jsp">
        <input name="code" type="text" size="28">
        <input type="submit" value="生成条形码">
</form>
```

（2）编写 result.jsp 页面，这个页面将接收首页中用户输入的产品编号，然后把产品编号传递给 ServletText 类，并指定不同的编码格式，在页面中显示各种编码的条形码，关键代码如下：

```
<tr align="center" bgcolor="#8AC52D">
    <td>Code 11 编码</td>
    <td>Code 128 编码</td>
    <td>Code 39 3:1 编码</td>
    <td>Code 39_2:1 编码</td>
</tr>
<tr align="center">
    <td><img src="ServletTest?code=${param.code }&type=<%=CodeType.Code_11%>"></td>
    <td><img src="ServletTest?code=${param.code }&type=<%=CodeType.Code_128%>"></td>
    <td><img src="ServletTest?code=${param.code }&type=<%=CodeType.Code_39_3_1%>"></td>
    <td><img src="ServletTest?code=${param.code }&type=<%=CodeType.Code_39_2_1%>"></td>
</tr>
<tr align="center" bgcolor="#8AC52D">
    <td>Code 93 编码</td>
    <td>Code 93 Extended 编码</td>
    <td>Ext Code 39 3:1 编码</td>
    <td>Ext Code 39 2:1 编码</td>
</tr>
```

（3）编写 CodeType 接口，在该接口中定义静态的常量，这些常量分别代表不同的编码格式，关键代码如下：

```
public interface CodeType {
    public static int Code_11=0x1;
    public static int Code_128=0x2;
    public static int Code_39_3_1=0x3;
    public static int Code_39_2_1=0x4;
    public static int Ext_Code_39_3_1=0x5;
    public static int Ext_Code_39_2_1=0x6;
    public static int Code_93=0x7;
    public static int Code_93_Extended=0x8;
    public static int InterLeaved_25_3_1=0x9;
    public static int InterLeaved_25_2_1=0x10;
    public static int MSI_model10_check=0x11;
    public static int Codabar_3_1=0x12;
    public static int Codabar_2_1=0x13;
    public static int EAN_13=0x14;
    public static int EAN_8=0x15;
}
```

（4）编写 ServletText 类，它是处理条形码生成请求的 Servlet，在该类中创建 JBarcodeBean 类的全局成员变量 barcode，它将被其他方法调用，关键代码如下：

```
public class ServletTest extends HttpServlet {
    JBarcodeBean barcode;
}
```

（5）编写 Servlet 初始化方法 init()，在该方法中初始化 JbarcodeBean 类的对象，并设置该对象生成条形码时显示条形码的文本信息，关键代码如下：

```
public void init(ServletConfig conf) throws ServletException {
    super.init(conf);
    barcode = new JBarcodeBean();                    //初始化条形码工具类
    barcode.setShowText(true);                       //设置生成的条形码显示文本信息
}
```

（6）编写 processRequest()方法，它通过 matchType()方法确定生成条形码的编码类型，然后设置生成条形码的其他参数信息，如生成条形码的产品编号、条形码的高度、编码类型等，最后将生成的条形码输出到客户端的输出流，关键代码如下：

```
protected synchronized void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        response.setContentType("image/gif");                //设置响应对象的数据类型为 GIF 图片
        OutputStream out = response.getOutputStream();       //获取响应对象的输出流
        BarcodeStrategy codeType = null;                     //创建条形码类型对象
        String typeStr = request.getParameter("type");
        int type = 0;
```

```
        if (typeStr == null || typeStr.isEmpty()) {
            type = Code_11;
        } else {
            type = Integer.parseInt(typeStr);
        }
        codeType = matchType(type);                              //匹配指定编码类型
        barcode.setCodeType(codeType);                           //设置指定编码类型
        barcode.setCode(request.getParameter("code"));           //设置条形码数值
        barcode.setBarcodeHeight(70);                            //设置条形码的高度
        barcode.setCheckDigit(false);
        barcode.setFont(barcode.getFont().deriveFont(14));       //设置文本字体
        barcode.gifEncode(out);                                  //输出条形码图像数据
    }
```

（7）编写 matchType()方法，该方法用于生成与页面请求和编码格式相匹配的条形编码对象，关键代码如下：

```
private BarcodeStrategy matchType(int type) {
    BarcodeStrategy codeType = null;
    switch (type) {
        //设置条形码的编码类型
        case Code_128:
            codeType = new Code128();
            break;
        case Code_39_3_1:
            codeType = new Code39();
            break;
        case Code_39_2_1:
            codeType = new Code39_2to1();
            break;
        case Ext_Code_39_3_1:
            codeType = new ExtendedCode39();
            break;
        case Ext_Code_39_2_1:
            codeType = new ExtendedCode39_2to1();
            break;
```

■ 秘笈心法

JbarcodeBean 组件是一个 JFC Swing 兼容的 JavaBean 组件，主要用来产生条形码，其下载网址为 http://jbarcodebean.sourceforge.net。它支持当前一些流行的条形码格式，如 Code 128、Code 39、Extended Code 39 等。本实例使用的 JbarcodeBean 组件的版本为 1.1.5。

第14章

图像操作

- 图片的大小
- 图片与鼠标相关的操作
- 图片与时间相关的操作
- 图片的动画效果
- 选择头像图片
- 图片的其他效果

14.1 图片的大小

在网站中浏览页面时往往会看到对图片放大或缩小的操作，尤其是一些图片资源的网站使用更是频繁，这些功能方便浏览者详细地看清楚图片，使网站更加人性化。

实例 373　打开自定义大小的图片　　初级

光盘位置：光盘\MR\14\373　　实用指数：★★★

实例说明

本实例通过一些简单的页面设计，使用户能够在文本框中输入图片的宽度和高度，当用户单击“预览”按钮后，程序会根据用户所指定的图片路径以及图片的大小在页面中显示该图片，运行结果如图 14.1 所示。



图 14.1　显示自定义大小的图片

关键技术

本实例主要在自定义的 JavaScript 函数中，应用 document 对象的 getElementById()方法获取文本框中输入的宽度和高度，并将这两个值设置为图片的高度和宽度，这样就实现了自定义大小图片的功能。

设计过程

（1）在 JSP 页面中设置页面的布局，关键代码如下：

```
<form name="form1" method="post" action="">
<p align="center">
 <input type="file" name="textfield">
</p>
<p align="center"><span class="style3">图片宽度：</span>
 <input name="widthtext" id="widthtext" type="text" value="" size="10" width="60"><br/>
          <span class="style3">图片高度：</span>
 <input name="heighttext" id="heighttext" type="text" size="10" width="60">
</p>
<p align="center">
      <input type="button" name="Submit2" value="预览" onClick="showpic();">
</p>
</form>
      <div id = 'div1' align="center">
         <img alt="" src="" id="img1">
      </div>
            </td>
         </tr>
      </table></td>
   </tr>
</table>
```

（2）编写 JavaScript 代码，用于获取文本框的高度和宽度的值，并设置为图片的宽度和高度，关键代码如下：

```
<script language="javascript">
         var str;
         function showpic(){
                  str = "";
                  var ima = form1.textfield.value;
                  var width=document.getElementById("widthtext").value
                  var height=document.getElementById("heighttext").value
                  document.getElementById("img1").src = ima;
                  document.getElementById("img1").width=width;
                  document.getElementById("img1").height=height;
```

```
        }
</script>
```

秘笈心法

在 JavaScript 中应用 document 对象的 getElementById()方法获取 HTML 页面元素的值或属性。

14.2　图片与鼠标相关的操作

实例 374	当鼠标经过图片时显示图片 光盘位置：光盘\MR\14\374	初级 实用指数：★★★

实例说明

在一些网站中，为了使页面更加美观，将网站工具栏中的按钮设置为模糊显示状态，当鼠标经过图片时，原本不清晰的图片变为清晰；当鼠标离开图片时，图片又回到不清晰的状态，运行结果如图 14.2 和图 14.3 所示。



图 14.2　鼠标未经过时的效果



图 14.3　鼠标经过时的效果

关键技术

在实现图片变化时，主要应用 onMouseover 和 onMouseout 鼠标事件，当鼠标经过图片时，会触发 onMouseover 事件，然后调用 JavaScript 函数改变图片 alpha 滤镜的透明度 opacity 属性，将 opacity 值设为 100 则表示完全不透明。当鼠标移出图片时，会触发 onMouseout 事件，再调用 JavaScript 函数改变图片 alpha 滤镜的透明度 opacity 属性值为 30。

设计过程

创建 index.jsp 页面，在 JavaScript 脚本中实现用于改变图片清晰的效果，关键代码如下：

```
<title>当鼠标经过图片时显示图片</title>
</head>
<SCRIPT language="JavaScript">
function makevisible(cur,which)
{
        if (which==0)
                cur.filters.alpha.opacity=100
        else
                cur.filters.alpha.opacity=30
}
</SCRIPT>
<body>
<div align="center">
<img src="magnifier.jpg" style="filter:alpha(opacity=20)" onMouseOver="makevisible(this,0)"
                onMouseOut="makevisible(this,1)" width="114" height="87">
</div>
```

秘笈心法

在 IE 中需要通过 filter 来定义透明度 opacity，而在 Mozilla 中可以直接解析 opacity，所以如果要使这个效果在两种浏览器中都得到支持，需要编写兼容两个浏览器的代码。

实例 375 当鼠标经过图像时给予文字提示

光盘位置：光盘\MR\14\375

初级

实用指数：★★★

实例说明

本实例实现的是当鼠标经过图像时，会给出一段预设的文字提示，而当鼠标离开时便没有文字提示。在浏览器不支持或禁止图像功能时就会直接给出预设的提示文字，运行效果如图 14.4 和图 14.5 所示。



图 14.4 鼠标经过图像时

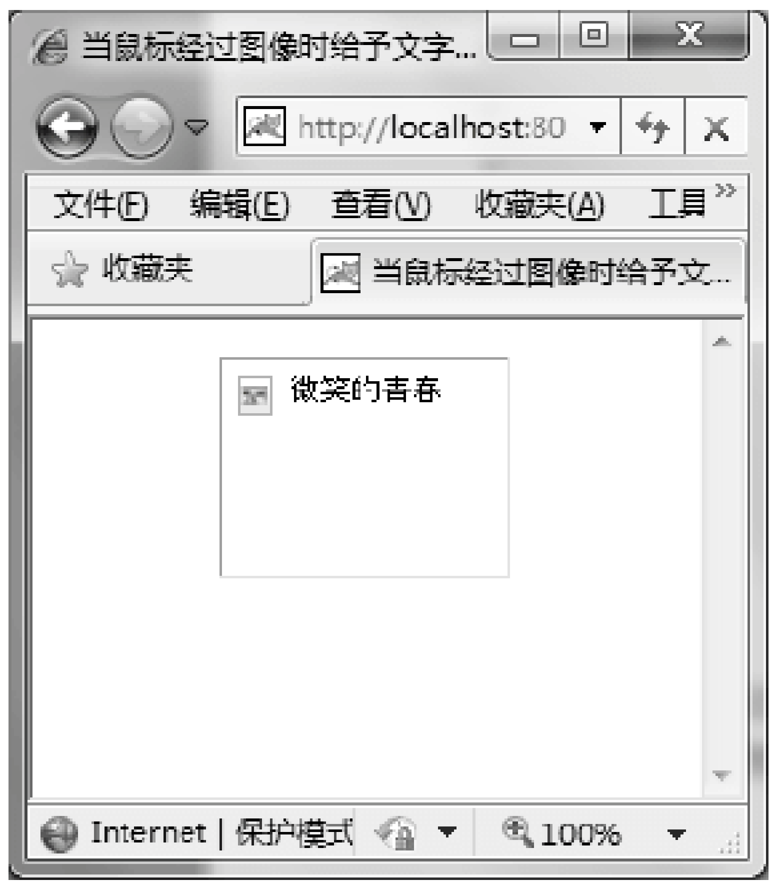


图 14.5 浏览器禁止或不支持时

关键技术

<img>标签的 Alt 属性用于当鼠标经过图片时显示的文字，这个属性值可以有也可以没有。Alt 属性还有一个用法，就是在不能显示图像的情况下，指定替代图像所使用的文字。

设计过程

创建 index.jsp 页面文件，主要代码如下：

```
<div align="center"><img src="smile.jpg"   alt="微笑的青春" width="114" height="87"></div>
```

秘笈心法

有时在打开浏览器时，会看不明白所显示的内容。这种现象是因为没有图像的替代文字，这时给图像加上说明文字就显得很有必要了。

实例 376 图片的预装载

光盘位置：光盘\MR\14\376

初级

实用指数：★★★

实例说明

本实例使用图像预装载原理，将图像在页面中以幻灯片的形式显示，实例运行结果如图 14.6 所示。

关键技术

实现本实例，主要是在页面的<div>标签中使用了 setInterval()函数，此函数用于在一定时间间隔内一直执行

同一段代码。应用该函数时，无须在自定义的 JavaScript 方法中调用，因为它是自动执行的，并且会一直执行下去。



（a）　　（b）　　（c）

图 14.6　幻灯片形式显示图像

设计过程

创建 index.jsp 页面，然后创建一个 photod 数组用于存放图像的名称，再创建一个用于存放图像对象的数组 photod2，在循环中使用 new Image()语句创建图像对象赋给 photod2 数组，把图像名称赋给每个图像对象的 src 属性，这样就将多个图像预装载到 cache 缓存中，关键代码如下：

```
<title>预装载图像制作幻灯片</title>
<script language="javascript">
	var j=0;
	var photod=new Array(8);
	for(var i=0;i<=8;i++){
		photod[i]=i;
	}
var photod2=[];
for(var i=0;i<photod.length;i++){
	photod2[i]=new Image();
	photod2[i].src=photod[i]+'.gif';
}
function showpic(){
	if(j==8)
		j=0;
	else;
		++j;
	var imagestr=photod2[j].src.split("/");
	var imagesrc=imagestr[imagestr.length-1];
	str="<img src='image/"+imagesrc+"'/>";
	div1.innerHTML=str;
}
</script>
</head>
<body onLoad="var begin=setInterval('showpic()',1000);">
<div id='div1' align="center"></div>
```

秘笈心法

setInterval()方法与 setTimeout()方法不同，它会在指定的时间间隔内反复执行某个 JavaScript 方法或某一段 JavaScript 脚本。与 setTimeout()方法类似的是，如果要终止执行，需要调用 clearInterval()方法。

实例 377	按时间随机变化的网页背景 光盘位置：光盘\MR\14\377	初级 实用指数：★★★

实例说明

本实例主要实现网页背景随机变化的功能。当打开该网页时，会每隔 5 秒显示不同的页面背景，运行结果

如图 14.7 所示。





图 14.7　每隔 5 秒换一次背景

关键技术

实现随机变化，主要应用 JavaScript 中用于产生随机数的 Math 类的 random()方法，该方法返回带正号的 double 类型的值，该值大于等于 0.0 且小于 1.0。返回值是一个伪随机选择的数，在该范围内（近似）均匀分布。

在 JavaScript 中，应用 document.body.background 来获取页面的背景属性，只要把图像 src 属性值赋值给背景属性即可。

设计过程

创建 index.jsp 页面文件，在本实例中使用 Math 类的 random()函数，随机产生 6 个数字并取整，然后使用 switch 语句根据当前随机产生的值设置背景图片，最后使用 setTimeout()函数每 5 秒调用一次 randompic()方法，主要代码如下：

```
</head>
<body onLoad="randompic();">
<center class="style2">后面管理员登录</center>
<table width="265" height="110" border="1" align="center" bordercolor="#6495ed">
  <tr>
    <td width="90"><span class="style5">用户名：</span></td>
    <td width="169"><input name="textfield" type="text"></td>
  </tr>
  <tr>
    <td><span class="style5">密  码：</span></td>
    <td><input name="textfield2" type="text"></td>
  </tr>
  <tr>
    <td colspan="2"><div align="center">
      <input name="Submit2" type="button" value="确定">
  
<input name="Button" type="button" value="取消">
    </div></td> </tr>
</table>
<p>
<script language="javascript">
function randompic(){
 var i = Math.floor(Math.random()*6);//取整并除 6
 var src = "";
 switch(i)       {
 case 0 :
         src = "Jellyfish.jpg";
         break;
 case 1:
         src = "Koala.jpg";
         break;
 case 2:
```

```
        src = "Lighthouse.jpg";
        break;
case 3:
        src = "Penguins.jpg";
        break;
case 4:
        src = "Tulips.jpg";
case 5:
        src ="Deset.jpg";
        break;
}
document.body.background=src;
setTimeout("randompic()",5000);
}
</script>
```

秘笈心法

在 JavaScript 中，由于调用 Math 类的 random()函数生成的随机数范围在 0.0～1.0 之间，都是小于 1 的小数，根据实际情况，如果想生成一些随机整数，可以将该函数的返回值乘以相应的整数值，然后再应用 Math 类的地板函数 floor()取整即可实现。

实例 378　左右循环滚动效果的图片

光盘位置：光盘\MR\14\378

初级

实用指数：★★★

实例说明

在设计网站时，为了美化页面效果，可以对图片进行无间断的循环滚动显示。当浏览者将鼠标经过滚动图片时，图片停止滚动，当浏览者将鼠标从图片上移开时，图片又继续活动，运行结果如图 14.8 所示。



图 14.8　无间断的图片循环滚动效果

关键技术

本实例主要使用 setTimeout()方法实现图片循环滚动效果，其语法格式如下：

```
setTimeout(function,milliseconds,[arguments])
```

参数说明

❶ function：要调用的 JavaScript 自定义的函数名或者 JavaScript 代码。

❷ milliseconds：设置超时时间，单位为毫秒。

设计过程

（1）编写 JavaScript 自定义函数，用于实现无间断的图片循环滚动效果，关键代码如下：

```
<SCRIPT language="javascript">
var n=4
//此变量控制图片移动的速度，图片越大滚动越快
td2.innerHTML=td1.innerHTML
var Mycheck;
function move(){
```

```
        if(td2.offsetWidth-div1.scrollLeft<=0)
                div1.scrollLeft-=td1.offsetWidth
        else{
                div1.scrollLeft++
        }
        Mycheck=setTimeout(move,n)
}
div1.onmouseover=function() {clearTimeout(Mycheck)}
div1.onmouseout=function() {Mycheck=setTimeout(move,n)}
move();
</SCRIPT>
```

（2）在页面的 div 层中实现图片的显示与隐藏，关键代码如下：

```
<DIV id=div1 style="OVERFLOW: hidden; WIDTH: 100%; COLOR: #ffffff">
        <TABLE cellSpacing=0 cellPadding=0 align=left border=0 cellspace="0">
          <TR>
            <TD id=td1 vAlign=top>
          <table width="876" height="73"   border="0" cellpadding="0" cellspacing="0">
              <tr>
                <td width="73" background="1.jpg"></td>
                <td width="73" background="2.jpg"></td>
                <td width="73" background="3.jpg"></td>
                <td width="73" background="4.jpg"></td>
                <td width="73" background="5.jpg"></td>
                <td width="73" background="6.jpg"></td>
                <td width="73" background="7.jpg"></td>
                <td width="73" background="8.jpg"></td>
                <td width="73" background="9.jpg"></td>
                <td width="73" background="10.jpg"></td>
                    <td width="73" background="11.jpg"></td>
                  <td width="73" background="12.jpg"></td>
              </tr>
            </table>
          </TD>
<TD id=td2 vAlign=top> </TD></TR></TABLE>
</DIV>
```

秘笈心法

实现本实例很简单，主要就是控制鼠标的 onMouseover 和 onMouseout 事件。页面在加载后，会利用 setTimeout()函数使图片循环滚动，当鼠标移到图片上时，会触发 onMouseover 事件调用 clearTimeout()函数终止图片滚动。

实例 379 浮动广告图片

光盘位置：光盘\MR\14\379

初级

实用指数：★★★

实例说明

在一些门户网站中，总会发现网页中的两侧有浮动的广告图片，无论怎么上下滚动页面图片都是保持在页面的最两侧。本实例用于实现在页面中放置浮动广告的功能，运行结果如图 14.9 所示。



图 14.9 浮动广告图片

关键技术

本实例主要应用了 body 标记中的 scrollLeft 属性、scrollTop 属性、offsetleft 属性和 offsetTop 属性来控制鼠标的跟随效果。下面对这几个属性进行说明。

- ❑ scrollLeft：设置或者获取对象左边界和窗口中目前可见内容的最左端之间的距离。
- ❑ scrollTop：设置或者获取对象最顶端和窗口中目前可见内容的最顶端之

间的距离。

- offsetLeft：获取对象相对于版面或由 offsetLeft 属性指定的父坐标的计算左侧位置。
- offsetTop：获取对象相对于版面或由 offsetTop 属性指定的父坐标的计算最顶端位置。

设计过程

（1）创建 index.jsp 页面，编写 JavaScript 脚本用于实现浮动图片的效果，关键代码如下：

```
<script language="JavaScript">
var delta=0.15
var layers;
function floaters() {
 this.items= [];
 this.addItem= function(id,x,y,content){
        document.write('<div  id='+id+'  style="z-index:  10;  position:  absolute;    width:80px;  height:60px;left:'+(typeof(x)=='string'?eval(x):x)+';top:'+(typeof(y) =='string'?eval(y):y)+'">'+content+'</div>');
        var newItem= {};
        newItem.object= document.getElementById(id);
        if(y>10) {y=0}
        newItem.x= x;
        newItem.y= y;
        this.items[this.items.length]= newItem;
 }
 this.play= function(){
        layers= this.items
        setInterval('play()',10);
 }
}
```

（2）编写自定义 div 标签以及函数位置，关键代码如下：

```
function play(){
 for(var i=0;i<layers.length;i++){
        var obj= layers[i].object;
        var obj_x= (typeof(layers[i].x)=='string'?eval(layers[i].x):layers[i].x);
        var obj_y= (typeof(layers[i].y)=='string'?eval(layers[i].y):layers[i].y);
        if(obj.offsetLeft!=(document.body.scrollLeft+obj_x)) {
                var dx=(document.body.scrollLeft+obj_x-obj.offsetLeft)*delta;
                dx=(dx>0?1:-1)*Math.ceil(Math.abs(dx));
                obj.style.left=obj.offsetLeft+dx;
        }
        if(obj.offsetTop!=(document.body.scrollTop+obj_y)) {
                var dy=(document.body.scrollTop+obj_y-obj.offsetTop)*delta;
                dy=(dy>0?1:-1)*Math.ceil(Math.abs(dy));
                obj.style.top=obj.offsetTop+dy;
        }
        obj.style.display= '';
 }
}
var strfloat = new floaters();
strfloat.addItem("followDiv",6,80,"<img src='hour.jpg'    border='0'>");
strfloat.play();
</script>
```

秘笈心法

在一些网站中，经常会加入一些浮动图片形式的广告，因此，对于本实例的实现也是很必要的。

实例 380	进度条的显示 光盘位置：光盘\MR\14\380	初级 实用指数：★★★

实例说明

在浏览网页时，经常会在载入图片时使用进度条，应用进度条可以使浏览者更清晰地了解网页内容的加载

进度，运行效果如图 14.10 所示。



图 14.10　40%的进度条

关键技术

本实例主要通过 document 对象的 getElementById()方法获取<span>标签对象，然后通过 prog.firstChild.nodeValue 语句指定<span>标签内部的值，prog.style.width 语句设置进度条的宽度，这个宽度是由变量 n 控制的，参数每次增长 10，直到 100 为止。实现进度条具有自动增长功能，可以应用 setTimeout()函数。

设计过程

（1）在网页中显示进度条的颜色效果，创建一个 CSS 文件设置进度条的样式，代码如下：

```
#test{
width:200px;
border:1px solid #000;
background:#fff;
color:#000;
}
#progress{
display:block;
width:0px;
background:#6594ed;
}
```

（2）在 index.jsp 页面中加载上述的 CSS 样式，并在<p>标签和<span>标签中使用，主要代码如下：

```
<div>开始上传进度为：</div>
<p id="test"><span id="progress">10%</span></p>
<script language="javascript">
progressTest(10);
</script>
```

（3）利用 JavaScript 脚本显示进度条上的百分比文本及进度条的进度，关键代码如下：

```
<script language="javascript">
function progressTest(n){
var prog=document.getElementById('progress');
prog.firstChild.nodeValue=n+"%";
prog.style.width=(n*2)+"px";
n+=10;
if(n>100){
 n=100;
}
setTimeout('progressTest('+n+')',1000);
}
</script>
```

秘笈心法

在 JavaScript 中，应用 document 对象的 getElementById()方法，根据 HTML 的元素指定的 id 值来获取某个元素对象。

实例 381　缩小与放大图片的效果

光盘位置：光盘\MR\14\381

初级

实用指数：★★★

实例说明

在页面中浏览图片时，有的图片是固定大小的，而且有的图片也不是很清晰，为了方便浏览图片，应该能够使图片具有放大缩小的功能。运行本实例，单击“放大”按钮时，图片会按照指定的大小不断增大，单击“缩小”按钮时，图片会按照指定的大小不断缩小，运行结果如图 14.11 所示。

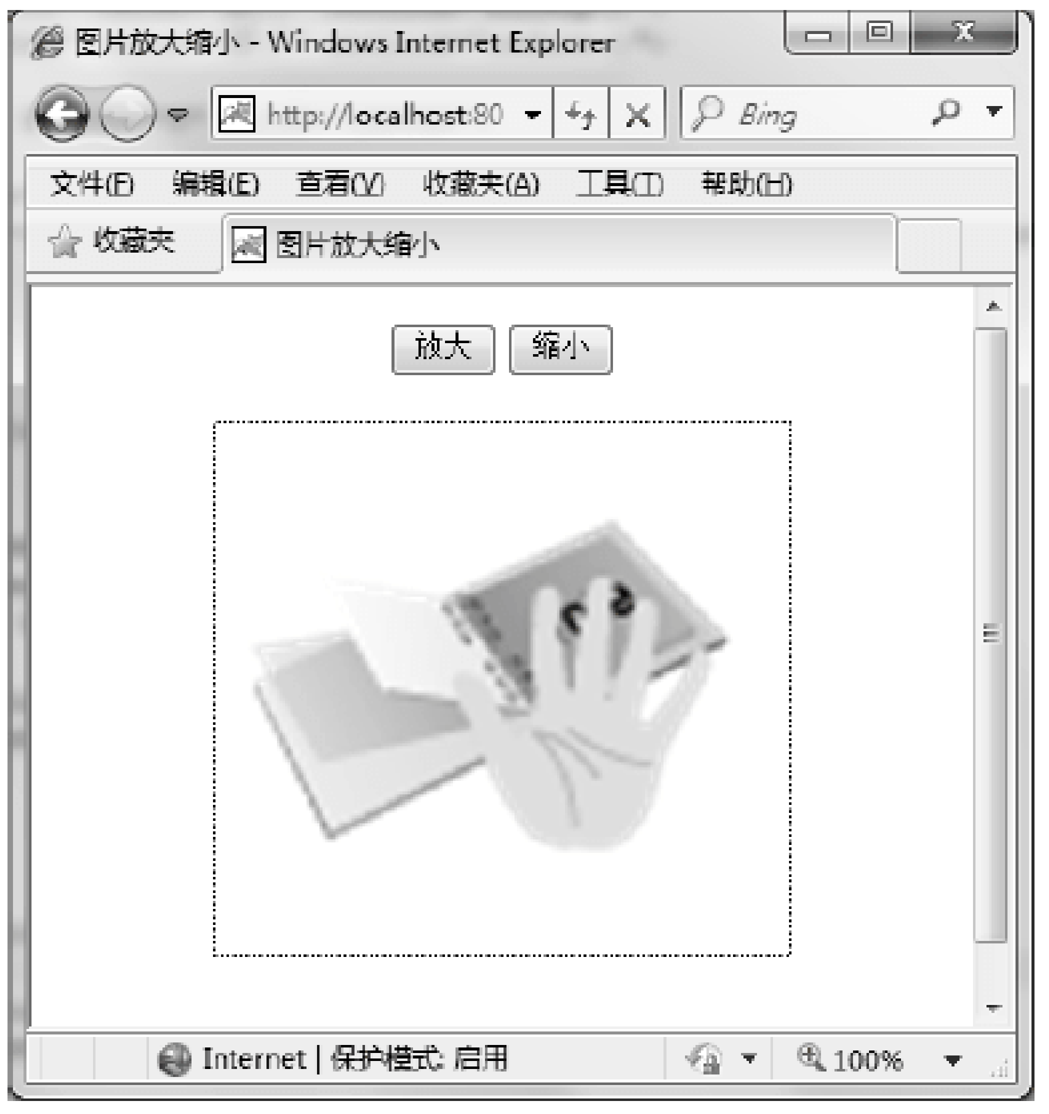


图 14.11　缩放图片

关键技术

实现本实例很简单，主要应用了一些简单的条件判断语句，然后根据当前图片的大小来放大或缩小图片的高度和宽度。

设计过程

（1）编写实现放大功能的 JavaScript 脚本代码。

```
function photoup()
{
	var height = images1.height;
	var width = images1.width;
	if ((height <= height * 2)||(width <= width * 2))
	{
		images1.height = images1.height + 40;
		images1.width = images1.width + 40;
	}
}
```

（2）编写实现缩小功能的 JavaScript 脚本代码。

```
function photoreduce()
{
	if ((images1.width > 100)||(images1.height > 100))
	{
		images1.height = images1.height - 40;
		images1.width = images1.width - 40;
	}
}
```

（3）在页面中调用两个自定义的放大与缩小功能的方法，关键代码如下：

```
<p align="center">
  <input type="button" value="放大" onclick="photoup()">
```

```
    <input type="button" value="缩小" onclick="photoreduce()">
</p>
```

秘笈心法

在本实例自定义的 JavaScript 方法 photoreduce()中，images.height 和 images.width 用于获取图像的高度和宽度，其中 40 为增长数，单位为像素。

实例 382 通过鼠标滚轮放大与缩小图片

光盘位置：光盘\MR\14\382

初级

实用指数：★★★

实例说明

本实例对实例 381 进行了优化，实现了使用鼠标滚轮放大与缩小图片。当鼠标移动到图片上时，图片将获取焦点，然后可以使用鼠标滚轮来改变图片的大小，如图 14.12 所示。

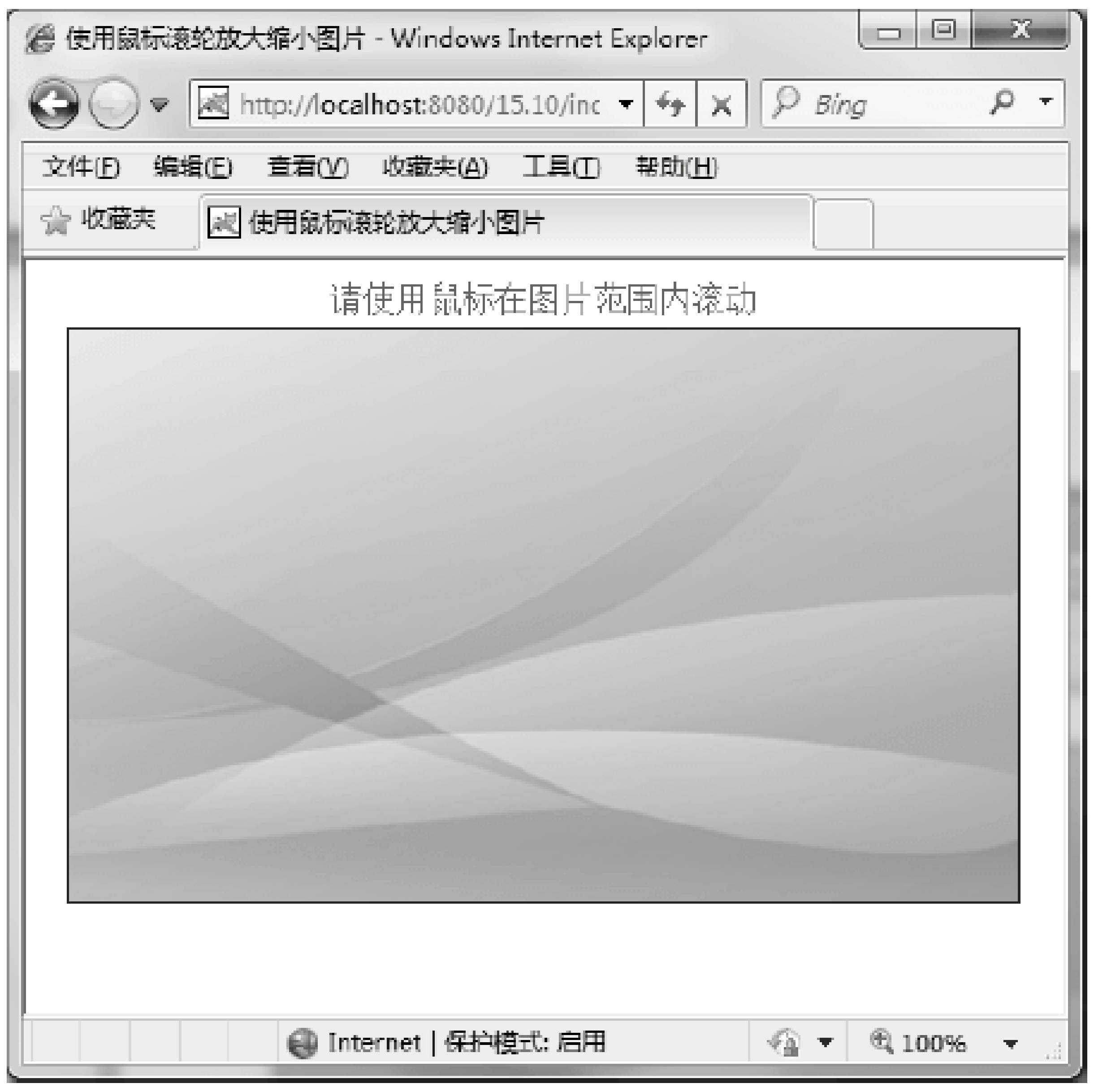


图 14.12　鼠标滚轮改变图片大小

关键技术

本实例主要是通过 event 对象的 wheelDelta 属性来获取鼠标的滚轮键的值，从而控制图片放大或缩小的倍数，再根据获取的值改变图片的 style 属性的 zoom 属性来实现。下面介绍 zoom 属性。

```
zoom:normal | number
```

参数说明

❶ zoom：设置获取对象的放大比例。

❷ normal：是使用对象的实际尺寸。

❸ number：百分数|无符号浮点实数，浮点实数值为 1.0 或百分数为 100%时相当于此属性的 normal 值。

设计过程

（1）编写 JavaScript 脚本，用于实现鼠标滚轮改变图片大小的方法，代码如下：

```
<script language="javascript">
function bigimg(i)
{
        var zoom = parseInt(i.style.zoom,10)||100;
        zoom += event.wheelDelta / 12;
        if(zoom > 0 )
                i.style.zoom=zoom+'%';
        return false;
```

```
}
</script>
```

（2）在页面中添加 OnMouseWheel 事件并调用 bigimg(this)方法，主要代码如下：

```
<table width="100%">
  <tr align="center">
    <td><span class="style1">请使用鼠标在图片范围内滚动</span></td>
  </tr>
</table>
<center>
<a href="http://www.mingrisoft.com">
<img src="bg.jpg" width="461" height="277" border="1" onmousewheel="return bigimg(this)">
</a>
</center>
```

秘笈心法

wheelDelta 是设置或获取滚轮滚动距离和方向的，而 OnMouseWheel 事件是当鼠标滚轮滚动时执行脚本（IE），如果是 Firefox 浏览器，则需要用 DommouseScroll 绑定滚轮事件。

实例 383	随鼠标移动的图片 光盘位置：光盘\MR\14\383	初级 实用指数：★★★

实例说明

在浏览一些网站时，经常会看到图标跟随鼠标的移动而移动，其中有些图标是 cur 鼠标文件，而有些却是图片。本实例将使用一个 GIF 格式的图片作为鼠标光标，然后通过 JavaScript 实现图片跟随鼠标移动。

关键技术

本实例将图片放在了一个层中，使层的位置与鼠标的位置相等，当层位置发生改变时，图片也会随之改变位置。

设计过程

（1）编写实现跟随鼠标移动图片的 JavaScript 脚本，代码如下：

```
<script language="javascript">
var x,y
function handlerM(){
 x =   window.event.x + document.body.clientLeft
 y =      window.event.y + document.body.clientTop;
}
function makes() {
 if (document.all) {
        var ob = document.all("tdiv")
        ob.style.posLeft=x
        ob.style.posTop=y
 }
        var timer=setTimeout("makes()",10)
}
document.onmousemove = handlerM;
</script>
```

（2）在页面中设置，初次加载时调用 makes()方法，代码如下：

```
<body onload="makes()" bgcolor="lightblue">
<div id="tdiv" style='position:absolute'>
 <img src='bs.jpg' width='32' height='32' class="spanstyle">
</div>
```

秘笈心法

本实例在<div>层中应用 style 样式的 position:absolute 属性，是用来固定<div>标签的，是相对位置，它是相

对于 body 位置而变化的。

实例 384　**左右拖动图片的效果**　初级

光盘位置：光盘\MR\14\384　实用指数：★★★

实例说明

本实例将使用 JavaScript 编写一个可以左右拖动的图片。当用户在图片上按住鼠标左键不放时，就可以左右拖动图片，当释放鼠标左键时，即将图片放置在释放鼠标左键时的位置上，如图 14.13 所示。

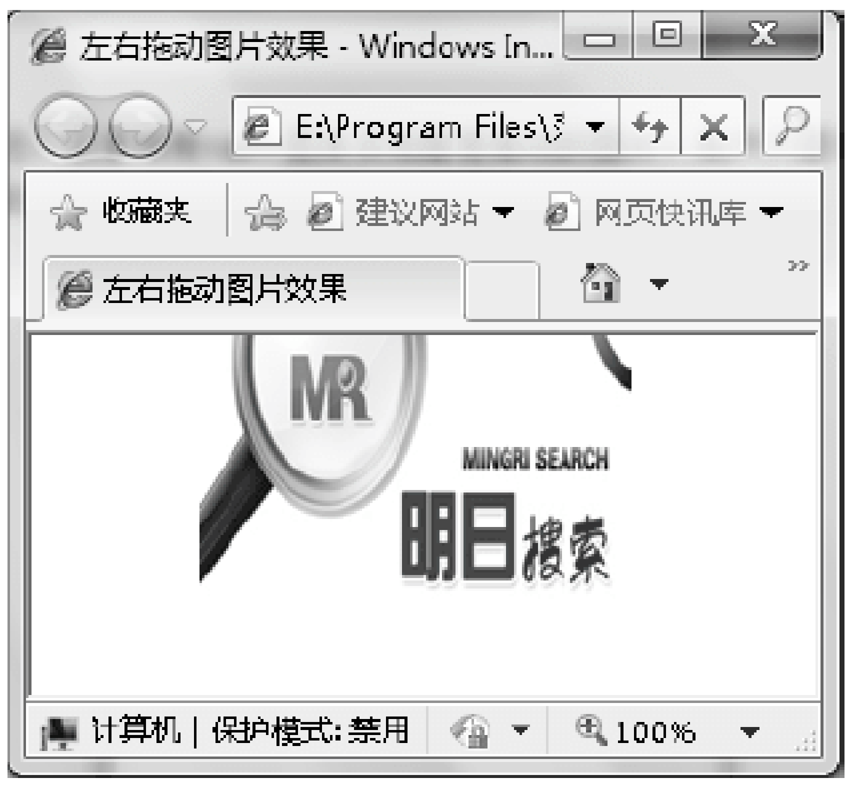


图 14.13　左右拖动图片的效果

关键技术

实现本实例，首先编写用于实现左右拖动图片的功能函数，再通过 window 对象的 setInterval()方法每隔 1ms 执行一次实现左右拖动图片的函数，最终可以实现左右拖动图片的效果。

设计过程

（1）编写实现可以左右拖动图片效果的 JavaScript 脚本，代码如下：

```
<Script language=JavaScript>
self.onError=null;
currentX = currentY = 0;
whichIt = null;
lastScrollX = 0; lastScrollY = 0;
NS = (document.layers) ? 1 : 0;
IE = (document.all) ? 1: 0;
function heartBeat()        {
        if(IE) {
                diffY = document.body.scrollTop; diffX = document.body.scrollLeft;
        }
        if(NS) {
                diffY = self.pageYOffset; diffX = self.pageXOffset;
        }
        if(diffY != lastScrollY) {
                percent = .1 * (diffY - lastScrollY);
                if(percent > 0) percent = Math.ceil(percent);
                else percent = Math.floor(percent);
                if(IE) document.all.floater.style.pixelTop += percent;
                if(NS) document.floater.top += percent;
                lastScrollY = lastScrollY + percent;
        }
        if(diffX != lastScrollX){
                percent = .1 * (diffX - lastScrollX);
                if(percent > 0)
                        percent = Math.ceil(percent);
                else
                        percent = Math.floor(percent);
                if(IE)
                        document.all.floater.style.pixelLeft += percent;
                if(NS)
                        document.floater.left += percent;
                lastScrollX = lastScrollX + percent;
        }
}
function checkFocus(x,y) {
            stalkerx = document.floater.pageX;
            stalkery = document.floater.pageY;
            stalkerwidth = document.floater.clip.width;
            stalkerheight = document.floater.clip.height;
            if( (x > stalkerx && x < (stalkerx+stalkerwidth)) && (y > stalkery && y < (stalkery+stalkerheight)))
```

```
                        return true;
            else
                        return false;
            }
function grabIt(e){
        if(IE) {
                whichIt = event.srcElement;
                while (whichIt.id.indexOf("floater") == -1) {
                        whichIt = whichIt.parentElement;
        if (whichIt == null){ return true; }
        }
                whichIt.style.pixelLeft = whichIt.offsetLeft;
                 whichIt.style.pixelTop = whichIt.offsetTop;
                currentX = (event.clientX + document.body.scrollLeft);
                currentY = (event.clientY + document.body.scrollTop); }
        else {
            window.captureEvents(Event.MOUSEMOVE);
            if(checkFocus (e.pageX,e.pageY)) {
                        whichIt = document.floater;
                        StalkerTouchedX = e.pageX-document.floater.pageX;
                        StalkerTouchedY = e.pageY-document.floater.pageY;
                }}
      return true;      }
function moveIt(e)   {
        if (whichIt == null) { return false; }
        if(IE) {
              newX = (event.clientX + document.body.scrollLeft);
              newY = (event.clientY + document.body.scrollTop);
              distanceX = (newX - currentX);       distanceY = (newY - currentY);
              currentX = newX;        currentY = newY;
              whichIt.style.pixelLeft += distanceX;
              whichIt.style.pixelTop += distanceY;
                if(whichIt.style.pixelTop < document.body.scrollTop) whichIt.style.pixelTop = document.body.scrollTop;
                if(whichIt.style.pixelLeft < document.body.scrollLeft) whichIt.style.pixelLeft = document.body.scrollLeft;
                if(whichIt.style.pixelLeft > document.body.offsetWidth - document.body.scrollLeft - whichIt.style.pixelWidth - 20) whichIt.style.pixelLeft
= document.body.offsetWidth - whichIt.style.pixelWidth - 20;
                if(whichIt.style.pixelTop > document.body.offsetHeight + document.body.scrollTop - whichIt.style.pixelHeight - 5) whichIt.style.pixelTop
= document.body.offsetHeight + document.body.scrollTop - whichIt.style.pixelHeight - 5;
                event.returnValue = false;}
        else {
                whichIt.moveTo(e.pageX-StalkerTouchedX,e.pageY-StalkerTouchedY);
            if(whichIt.left < 0+self.pageXOffset) whichIt.left = 0+self.pageXOffset;
            if(whichIt.top < 0+self.pageYOffset) whichIt.top = 0+self.pageYOffset;
            if( (whichIt.left + whichIt.clip.width) >= (window.innerWidth+self.pageXOffset-17)) whichIt.left = ((window.innerWidth+self.pageXOffset)
-whichIt.clip.width)-17;
            if( (whichIt.top + whichIt.clip.height) >= (window.innerHeight+self.pageYOffset-17)) whichIt.top = ((window.innerHeight+self.pageYOffset)
-whichIt.clip.height)-17;
            return false;}
      return false;      }
function dropIt()      {
        whichIt = null;
      if(NS) window.releaseEvents (Event.MOUSEMOVE);
      return true;
}
if(NS) {
        window.captureEvents(Event.MOUSEUP|Event.MOUSEDOWN);
        window.onmousedown = grabIt;
        window.onmousemove = moveIt;
        window.onmouseup = dropIt;
}
if(IE) {
        document.onmousedown = grabIt;
        document.onmousemove = moveIt;
        document.onmouseup = dropIt;
}
if(NS || IE)
        action = window.setInterval("heartBeat()",1);
</Script>
```

（2）在<body>标签中添加一段 CSS 样式，主要代码如下：

```
<STYLE type=text/css>#floater {
 LEFT: 445px; POSITION: absolute; TOP: -3px; VISIBILITY: visible; WIDTH: 125px; Z-INDEX: 10}
</STYLE>
```

（3）页面调用代码如下：

```
<DIV align=center id=floater>
 <TABLE height=10 width=24>
       <TBODY>
             <TR>
                   <TD align=middle height=6 vAlign=center width=76>
                         <img src="mr.jpg" width="170" height="113"    border=0 style="cursor:pointer">
                   </TD>
             </TR>
       </TBODY>
 </TABLE>
</DIV>
```

秘笈心法

本实例自定义了一个 CSS 样式 floater，并在 JavaScript 中多次调用此方法，如在 checkFocus(x)获取焦点时使用了 document 对象获取 floater.pageX 的值。

实例 385	随意拖动图片 光盘位置：光盘\MR\14\385	初级 实用指数：★★★

实例说明

本实例主要使用<div>层来实现随意拖动的图片。用户只需在页面中单击图片，然后移动鼠标，便可将图片从当前位置移动到另一个位置上，如图 14.14 所示。



图 14.14　随意拖动的图片

关键技术

本实例通过自定义一个 JavaScript 函数，模拟鼠标的相关操作，然后将对应的函数值赋值给页面，使用页面中的鼠标事件调用自定义的函数，使用 event 对象的 x 和 y 属性获取当前鼠标的位置。

设计过程

（1）声明一个自定义函数 mousedown()，使用变量 x1 和 y1 获取鼠标在层左上角的距离，代码如下：

```
<script language="javascript">
drag = 0;
move = 0;
function mousedown(){
 if(drag)       {
       X1 = window.event.x - parseInt(dragimages.style.left);
       Y1 = window.event.y - parseInt(dragimages.style.top);
```

```
        dragimages.style.Index += 1;
        move = 1;
}}
function mouseStop(){
window.event.returnValue = false;
}
function mousemove(){
if (move)      {
        dragimages.style.left = window.event.x - X1;
        dragimages.style.top = window.event.y - Y1;
}}
function mouseup(){
move = 0;
}
```

（2）在 id 为 div1 层的 OnMouseMove 事件中调用 mousemove()方法，在 OnMouseDown 事件中调用 mousedown()方法，在 OnMouseUp 事件中调用 mouseup()方法，在 OnDragStart 事件中调用 mouseStop()方法，代码如下：

```
function remove(){
document.all.div1.onmousemove = mousemove;
document.all.div1.onmousedown = mousedown;
document.all.div1.onmouseup = mouseup;
document.all.div1.ondragstart   = mouseStop;
}
</script>
```

（3）在页面中加载调用 remove()方法，代码如下：

```
<body onLoad="remove()">
<div  id="div1"  onMouseOver="dragimages=div1;drag=1;"style="height:77px;left:10px;position:absolute;top:10px;width:90px"><img  name="img1"
alt="" border="2"   src="mr.jpg" style="cursor:pointer ">
</div>
<p>
```

秘笈心法

本实例主要自定义 4 个鼠标动作函数 mousemove()、mousedown()、mouseup()和 mouseStop()，分别实现鼠标移动时、单击时、松开时和无动作时的应用，并用 event 对象获取鼠标的位置与图片层中当前对象的定位实现图片移动的功能。

实例 386	改变图片获取焦点时的状态 光盘位置：光盘\MR\14\386	初级 实用指数：★★★

实例说明

在开发网站时，为了突出某个图片或者链接的焦点，需要设置其获取焦点时的显示状态。在本实例中，当浏览者的鼠标移动到图片上时，图片会以灰度状态显示，如图 14.15 和图 14.16 所示。



图 14.15　图片获取焦点时状态 1



图 14.16　图片获取焦点时状态 2

关键技术

实现本实例，主要使用了 CSS 样式中的 opacity 滤镜，通过设置滤镜中的透明度来改变图片获取焦点时的显示状态。

设计过程

（1）实现改变图片获取焦点时显示状态的 JavaScript 脚本，代码如下：

```
<script language="JavaScript">
function visible(cursor,i){
if (i==0)
 cursor.filters.alpha.opacity=100;
else
 cursor.filters.alpha.opacity=30;
}
</script>
```

（2）在<body>中的页面代码使用了 onMouseOver 事件和 onMouseOut 事件调用 visible()方法，主要代码如下：

```
<table border="0" cellpadding="0" cellspacing="0">
  <tr>
    <td align="center" bgcolor="#CCCCCC">
 <a class="link_image" href="http://www.mingrisoft.com">
 <img src="key.jpg" border="0" style="filter:alpha(opacity=100)" onMouseOver="visible(this,1)" onMouseOut="visible(this,0)" width="148"
height="121">
 </a>
 </td>
</tr>
</table>
```

秘笈心法

alpha 滤镜效果还有如下属性：finishOpacity 用于设置渐变透明度；style 用于设置透明区域的形状，0 表示统一形式，1 表示线形，2 表示放射状，3 表示长方形。startX 和 startY 用于设置透明度的开始的横、纵坐标。finishX 和 finishY 用于设置透明度的结束横、纵坐标。

实例 387	抖动的图片 光盘位置：光盘\MR\14\387	初级 实用指数：★★★

实例说明

在浏览一些网站时，经常会看到带有抖动效果的图片，将鼠标移动到图片上时，此图片就开始抖动；当鼠标离开图片时，图片又恢复到原来的静止状态，实例运行效果如图 14.17 所示。



图 14.17 抖动效果的图片

关键技术

本实例主要利用 if…else 语句判断当前的变量 a 的值，如果 a 的值符合某一条件，则执行图片向某一方向的抖动方法如下。

使图片向上抖动：

```
shake.style.top = parseInt(shake.style.top)+dithering;
```

使图片向左抖动：

```
shake.style.left = parseInt(shake.style.left)+dithering;
```

设计过程

（1）JavaScript 脚本的 shake.style.top、shake.style.left 分别使图片向上、向左抖动，定义了一个抖动频率的变量 dithering，此值越大抖动的频率就越快，主要代码如下：

```
<script language="javascript">
var dithering = 10;//抖动频率
var stopit = 0;
var a=1;
function init(which){
 stopit = 0;
 shake = which;
 shake.style.left = 0;
 shake.style.top = 0;
}
function ditheringimage(){
 if ((!document.all && !document.getElementById)||stopit ==1)
 return
 if (a==1)      {
        shake.style.top = parseInt(shake.style.top)+dithering;
 }else if (a==2)
 {
        shake.style.left = parseInt(shake.style.left)+dithering;
 }
 else if (a==3){
        shake.style.top = parseInt(shake.style.top)-dithering;
 }
 else{
        shake.style.left = parseInt(shake.style.left)-dithering;
 }
 if (a<4)
 a++;
 else
 a=1;
 setTimeout("ditheringimage()",50);
}
function stoprattle(which){
 stopit = 1;
 which.style.left = 0;
 which.style.top = 0;
}
</script>
```

（2）添加 CSS 样式，代码如下：

```
<style>
.ditheringimg
{
 position : relative
}
</style>
```

（3）在页面中添加图片，并通过 onMouseOver 事件和 onMouseOut 事件调用 JavaScript 定义的 ditheringimage()和 stoprattle()方法，代码如下：

```
<div align="center">
<img src="afraid.jpg" border="0" class="ditheringimg" onMouseOver="init(this);ditheringimage()" onMouseOut="stoprattle(this)">
</div>
```

秘笈心法

CSS 样式中的 position:relative 属性表示相对定位，是相对于容器定位，对象不可重叠，但将依据 left、right、top、button 等属性在文档中偏移位置。

实例 388　鼠标移动放大图片

光盘位置：光盘\MR\14\388

初级

实用指数：★★★

实例说明

在动态网站中，经常会用到图片。如果要以图片的实际尺寸进行显示，将会给整个网站带来不便，而且图片大小不好控制，使网站页面布局混乱。要解决此问题，可以在页面中加入相关鼠标事件。运行本实例，网页中的图片以缩略图的形式显示，当鼠标移动到指定图片时，该图片将以实际尺寸进行显示；当鼠标离开图片时，该图片又将以缩略图的形式显示，实例运行结果如图 14.18 和图 14.19 所示。



图 14.18　图片放大前



图 14.19　图片放大后

关键技术

本实例主要应用 JavaScript 中的 Math 对象实现鼠标移动放大，此对象是一个静态对象，不能使用 new 关键字创建对象实例，应该直接使用"对象名.成员"的形式来访问其属性或方法。

Math 的语法格式如下：

```
Math.mdethod(value[.value])
```

Math 对象的相关属性及说明如表 14.1 所示。

表 14.1　Math 对象的相关属性及说明

属　性	说　明
E	欧拉常量
LN2	2 的自然对数
LN10	10 的自然对数
LOG2E	以 2 为底数的 e 的对数
LOG10E	以 10 为底数的 e 的对数
PI	圆周率常数
SQRT1/2	0.5 的平方根
SQRT2	2 的平方根

Math 对象的相关方法及说明如表 14.2 所示。

表 14.2　Math 对象的相关方法及说明

方　法	说　明
Abx(x)	返回 x 的绝对值
Acos(x)	返回 x 弧度的反余弦
Asin(x)	返回 x 弧度的反正弦
Atan(x)	返回 x 弧度的反正切
Atan2(x,y)	返回坐标(x,y)对应的极坐标角度
Ceil(x)	返回大于或等于 x 的最小整数
Cos(x)	返回 x 的余弦
Exp(x)	返回 e 的 x 乘方
Floor(x)	返回小于或等于 x 的最大整数
Log(x)	返回 x 的自然对数
Max(x,y)	返回 x 和 y 中的最大数
Min(x,y)	返回 x 和 y 中的最小数
Pow(x,y)	返回 x 对 y 的次方
Random()	返回 0 和 1 之间的随机数
Round(x)	返回最接近 x 的整数
Sin(x)	返回 x 的正弦值
Sqrt(x)	返回 x 的平方根
Tan(x)	返回 x 的正切值

设计过程

（1）利用 JavaScript 脚本创建 mousemove()和 mouseout()函数，mousemove()主要显示图片的实际大小，mouseout()主要用于当鼠标离开图片时，图片将以原始大小形式显示，代码如下：

```
<script language="javascript">
var w=images1.width;
var h=images1.height;
images1.height = Math.floor(h*0.5);
images1.width = Math.floor(w*0.5);
function mousemove()
{
        images1.height = h;
        images1.width = w;
}
function mouseout()
{
        images1.height = Math.floor(h*0.5);
        images1.width = Math.floor(w*0.5);
}
</script>
```

（2）将插入图片的 border 属性设置为 0，同时当鼠标在图片上经过时，图片的 onMouseMove 事件调用自定义函数 mousemove()，而当鼠标移出图片时，应用图片的 onMouseOut 事件就会调用 mouseout()函数，关键代码如下：

```
<input NAME="images1" TYPE="image" ID="images1" SRC="manager.jpg"
 ALIGN="MIDDLE" BORDER="0"
  onMouseMove=mousemove(); onMouseOut=mouseout();>
```

秘笈心法

本实例应用了 JavaScript 中 Math 类的 floor()函数，该函数用于返回小于或等于某个数的最大整数，被称为

"地板函数"。例如，数字 1.5 在应用 floor()函数之后，返回值是 1，而-1.5 返回值则为-2。

14.3　图片与时间相关的操作

实例 389	定时隐藏图片 光盘位置：光盘\MR\14\389	初级 实用指数：★★★

实例说明

本实例实现了定时隐藏图片的功能，类似于倒计时功能，当页面被加载时程序会自动倒数 10 秒开始计时，当倒计时为 0 时，则设置层中的 visibility 属性为隐藏状态。由于图片在层中，所以图片同时也被隐藏，运行结果如图 14.20 所示。



图 14.20　定时隐藏图片

关键技术

本实例首先将图片放置在层中，然后将层的 visibility 属性设置为 hidden，使层被隐藏，由于图片放置在层中，所以图片同时也被隐藏。

设计过程

（1）实现预设时间隐藏图片的 JavaScript 脚本代码如下：

```
<SCRIPT LANGUAGE="JavaScript">
var sec=10;
var timer;
function hidepic()
{
 sec--;
 if (sec==0){
        textfield.value = "图片已隐藏";
        soccer.style.visibility =(soccer.style.visibility == "hidden") ? "visible" : "hidden";
 }
 else{
        textfield.value = "图片会在  "+sec+"  秒后隐藏";
        setTimeout("hidepic()",1000);
 }
}
</SCRIPT>
```

（2）当页面被加载时调用 hidepic()函数，关键代码如下：

```
<center>
      <input name="textfield" type="text" size="20"> <br>
```

```
<DIV ID="soccer"   align="center">
<img border="0" src="view.jpg" >
</DIV>
</center>
```

秘笈心法

对于本实例定时隐藏图片的效果，在网站中的应用十分广泛，通过 visibiliy 属性可以实现图片的隐藏与显示，由此也可通过获取系统时间来设置图片在指定时间内隐藏，然后在指定的时间内再次显示，用法十分灵活，原理都是一样的。

实例 390　根据时间变换页面背景

初级

光盘位置：光盘\MR\14\390

实用指数：★★★

实例说明

有时为了丰富页面的显示效果，将页面制作成根据时间变换页面背景的样式，这样浏览者对此网站不会感觉厌倦，同时也会觉得网站制作得非常新颖。本实例通过 Date 对象的 getHours()方法获得当前系统时间的小时，然后根据不同的时间段来改变页面的背景图片，效果如图 14.21 所示。

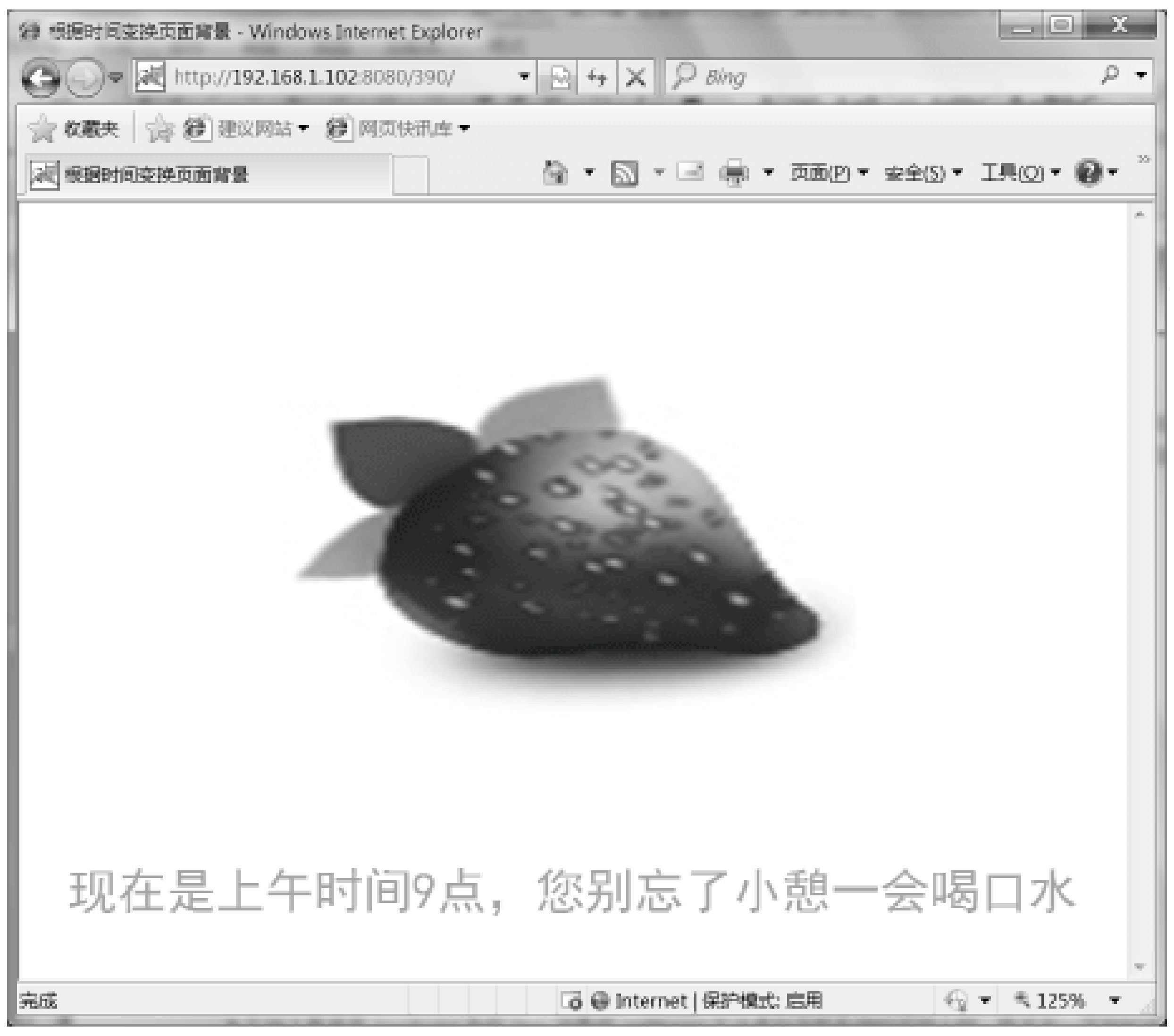


图 14.21　上午时间变换的背景

关键技术

本实例主要使用 JavaScript 中 Date 对象的 getHours()方法得到当前系统时间的小时，然后在一定的时间段内判断当前小时是否符合指定的时间段，如果符合则使用 document 对象的 write()方法在页面中显示一幅图片并在图片下方输出指定的提示信息。

设计过程

（1）使用 now.getHours()函数获取当前系统时间的小时，根据此时间变换不同的背景，主要 JavaScript 脚本代码如下：

```
<script language="javascript">
var now = new Date();
var hour = now.getHours();
if (hour >= 0 && hour <5){
        document.write("<center><img src='1.jpg' width='600' height='399'><center>");
        document.write("<p>");
        document.write("<font size = 6 face = 黑体 color =#ff9900 >现在是凌晨时间"+hour+"点，祝您好梦</font>");
}
if (hour >= 5 && hour <8){
```

```
        document.write("<center><img src='2.jpg' width='600' height='399'><center>");
        document.write("<p>");
        document.write("<font size = 6 face = 黑体 color =#ff9900 >现在是早上时间 "+hour+"点，祝您一天心情愉快</font>");
}
if (hour >= 8 && hour <11){
        document.write("<center><img src='3.jpg' width='600' height='399'><center>");
        ocument.write("<p>");
        document.write("<font size = 6 face = 黑体 color =#ff9900 >现在是上午时间"+hour+"点，您别忘了小憩一会喝口水</font>");
}
if (hour >= 11 && hour <13){
        document.write("<center><img src='4.jpg' width='600' height='399'><center>");
        document.write("<p>");
        document.write("<font size = 6 face = 黑体 color =#ff9900 >现在是中午时间"+hour+"点,记得要多吃饭 </font>");
}
if (hour >= 13 && hour < 17){
        document.write("<center><img src='5.jpg' width='600' height='399'><center>");
        document.write("<p>");
        document.write("<font size = 6 face = 黑体 color =#ff9900 >现在是下午时间"+hour+"点，久坐办公室要适当起身运动一下</font>");
}
if (hour >= 17 && hour < 24){
        document.write("<center><img src='6.jpg' width='600' height='399'><center>");
        document.write("<p>");
        document.write("<font size = 6 face = 黑体 color =#ff9900>现在是晚上时间"+hour+"点，要注意早点入睡</font>");
}
</script>
```

（2）添加一段 CSS 样式，代码如下：

```
<style type="text/css">
body {
        background-color: #FFFFFF;
}
</style>
```

秘笈心法

JavaScript 中的 Date 对象的 getHours()方法返回的是小时，返回值是一个数字，在 0～23 之间，表示包含或开始于此 Date 对象表示的瞬间的一天中的小时（用本地时区进行解释）。

实例 391 使图片不停闪烁

光盘位置：光盘\MR\14\391

初级 实用指数：★★★

实例说明

在一些招商的网站中，页面中包含着大量的广告信息，有时会使用一些网页特效来吸引浏览者。本实例为了使图片链接更具有吸引力，在图片中增加了不停闪烁的效果，运行结果如图 14.22 所示。



图 14.22　不停闪烁的图片

关键技术

本实例主要是对层进行操作，利用 glint()函数对三目运算符设置图片的隐藏与显示状态，并添加 setTimeout()在预设时间内重复执行此操作，从而实现图片闪烁的效果。

设计过程

（1）编写 JavaScript 脚本代码，使用 setTimeout()调用 glint()方法，设置图片的显示与隐藏状态，代码如下：

```
<SCRIPT LANGUAGE="JavaScript">
var counter = 0;
function soccerOnload()
```

```
{
        setTimeout("glint()", 200);
}
function glint()
{
        div1.style.visibility = (div1.style.visibility == "hidden") ? "visible" : "hidden";
        counter += 1;
        setTimeout("glint()", 200);
}
</SCRIPT>
```

（2）在 index.jsp 页面中把图片加入层中用于设置显示和隐藏，代码如下：

```
<body onload="soccerOnload();">
        <div ID="div1" STYLE="position: absolute; left: 150; top: 0">
                <a href="http://www.mingribook.com" target="_blank">
                        <p></p> <img name="image1" src="mr.jpg">
                        <p></p> <font size="3pt" color="lightblue"> 最新上市图片 </font> </a>
        </div>
</body>
```

秘笈心法

三目运算符的语法格式如下：

```
<表达式 1>?<表达式 2>:<表达式 3>;
```

其中，“?”运算符的含义是：如果表达式 1 的值为真，则求表达式 2 的值并把它作为整个表达式 1 的值；如果表达式 1 的值为假，则求表达式 3 的值并把它作为整个表达式 1 的值。

实例 392	上下跳动的图片 光盘位置：光盘\MR\14\392	初级 实用指数：★★★

实例说明

在浏览网站的过程中，经常会看到页面中漂浮的或者上下跳动的图片链接。本实例将使用 JavaScript 实现一个上下跳动的图片。在页面加载后，图片将会从左向右上下跳动，当图片移动到页面的左侧时，图片又会从左侧向右移动，如图 14.23 所示。

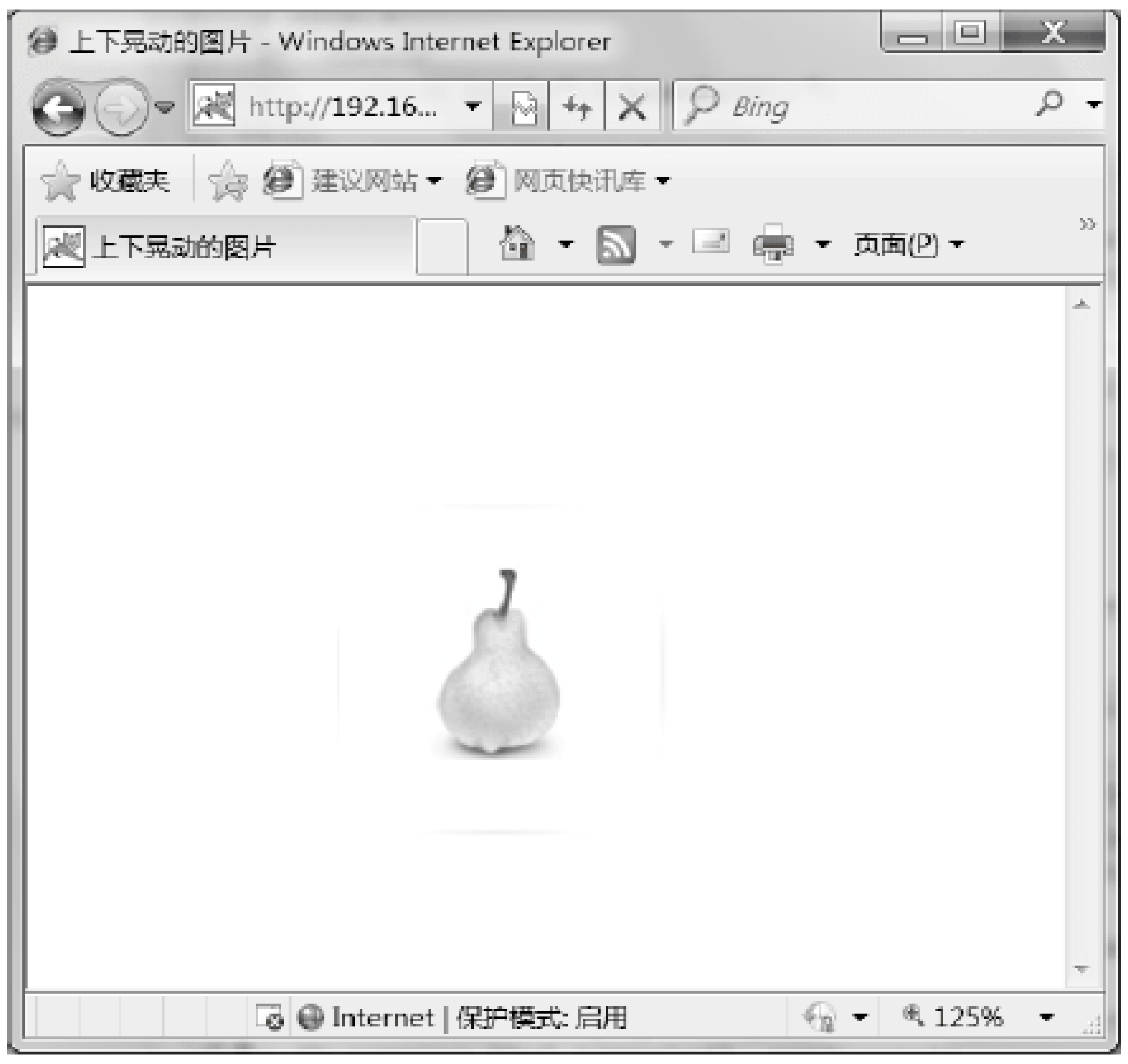


图 14.23　上下跳动的图片

关键技术

本实例中的技术要点是实现图片上下跳动，上下跳动的图片在屏幕中有一定的限制，当图片向下移到指定位置时，图片将会迅速反弹，然后继续跳动。

设计过程

（1）实现图片上下跳动的 JavaScript 代码如下：

```
<script language="JavaScript">
var ballWidth = 40;
var ballHeight = 40;
var BallSpeed = 10;
var maxBallSpeed = 50;
var xMax;
var yMax;
var xPos = 0;
var yPos = 0;
var xDir = 'right';
var yDir = 'down';
var superballRunning = true;
var tempBallSpeed;
var currentBallSrc;
var newXDir;
var newYDir;
function initializeBall() {
        xMax = document.body.clientWidth
        yMax = document.body.clientHeight
        document.all("superball").style.visibility = "visible";
        setTimeout('moveBall()',1000);
}
function moveBall() {
    if (superballRunning == true) {
        calculatePosition();
        document.all("superball").style.left = xPos + document.body.scrollLeft;
        document.all("superball").style.top = yPos + document.body.scrollTop;
        setTimeout('moveBall()',50);
    }
}
function calculatePosition() {
        if (xDir == "right") {
                if (xPos > (xMax - ballWidth - BallSpeed)) {
                        xDir = "left";
                }
        }
        else if (xDir == "left") {
                if (xPos < (0 + BallSpeed)) {
                        xDir = "right";
                }
        }
        if (yDir == "down") {
                if (yPos > (yMax - ballHeight - BallSpeed)) {
                        yDir = "up";
                }
        }
        else if (yDir == "up") {
                if (yPos < (0 + BallSpeed)) {
                        yDir = "down";
                }
        }
        if (xDir == "right") {
                xPos = xPos + BallSpeed;
        }
        else if (xDir == "left") {
                xPos = xPos - BallSpeed;
        }
        else {
                xPos = xPos;
        }
        if (yDir == "down") {
                yPos = yPos + BallSpeed;
        }
        else if (yDir == "up") {
                yPos = yPos - BallSpeed;
```

```
        }
        else {
                yPos = yPos;
        }
}
window.onload = initializeBall;
window.onresize = new Function("window.location.reload()");
</script>
```

（2）在页面中添加 CSS 样式，代码如下：

```
<style type="text/css">
#superball {
        position:absolute;
        left:0;
        top:0;
        visibility:hide;
        visibility:hidden;
        width:40;
        height:40;
}
</style>
```

（3）在页面中编写一个<span>标记，用此来引用插入 CSS 样式，并把图片放在层中，代码如下：

```
<p><a href="http://www.mingrisoft.com" style="cursor:pointer ">
</a></p>
<a href="http://www.mingribook.com" style="cursor:pointer ">
<span id="superball">
 <img name="superballImage" src="fb.jpg" height="150" width="150" border="0">
</span>
</a>
```

秘笈心法

本实例主要是在 JavaScript 中，应用 scrollLeft 和 scrollTop 属性动态修改图片相对页面的位置来实现的。

实例 393　**左右晃动的图片**

光盘位置：光盘\MR\14\393

初级

实用指数：★★★

实例说明

本实例使用 JavaScript 实现了一个能够左右晃动的图片，当页面被加载后，图片就会从左向右移动，在移动到一定范围内后又开始向左移动，如此反复执行，实例运行结果如图 14.24 所示。



图 14.24　左右晃动的图片

关键技术

本实例主要通过 JavaScript 自定义的两个函数对图片的位置进行移动，rect()函数用于指定图片位置，move()函数用于设置图片移动的范围。然后应用 setTimeout()方法，每隔 10 毫秒调用一次 move()函数，从而实现左右无间断晃动的效果。

设计过程

（1）使用 JavaScript 脚本实现左右晃动效果的图片，关键代码如下：

```
function move(picobj,picmovewidth,c)
{
  window_width = document.body.offsetWidth;
  imgobj = document.images[picobj];
  image_width = imgobj.width;
  x1 = imgobj.style.left;
  x = Number(x1.substring(0,x1.length-2));
  if (c == 0) {
        if (picmovewidth == 0) {
            right_margin = window_width - image_width;
           rect(x,right_margin,1);
        }
        else {
                right_margin = x + picmovewidth - image_width;
                if (picmovewidth < x + image_width)
                        window.alert("No space for moving!");
                else
                        rect(x,right_margin,1);
        }
    }
   else {
          if (picmovewidth == 0)
                right_margin = window_width - image_width;
        else {
               x = Math.round((window_width-picmovewidth)/2);
                right_margin = Math.round((window_width+picmovewidth)/2)-image_width;
        }
        rect(x,right_margin,1);
    }
}
</script>
```

（2）在页面中添加要左右晃动的图片，然后再利用 setTimeout()方法每隔 10 毫秒调用一次 move()函数，代码如下：

```
<a href="http://www.mingrisoft.com">
<img src="mrbook.jpg" width=170 height=113 name="picture" style='position: absolute; top: 10px; left: 30px;' border=0>
</a>
   <script language="JavaScript">
        setTimeout("move('picture',600,1)",10);
   </script>
```

秘笈心法

本实例用到了 JavaScript 中 String 类的 subString(startIndex,endIndex)函数，应用它可以截取某个字符串的子串，截取时是从 startIndex 位置开始截取到 endIndex 位置（不包括 endIndex 位置的字符），如果不指定 endIndex 参数，则会一直截取到此字符串的末尾。

实例 394	移动变形的图片 光盘位置：光盘\MR\14\394	初级 实用指数：★★★

实例说明

本实例实现了在页面中飘动的图片及变形效果，在页面被加载时，图片会随意飘动，并且在飘动的过程中改变图片本身的形状，当飘动的图片移动到页面的边缘时，图片会马上反弹到页面中继续移动，效果如图 14.25 和图 14.26 所示。

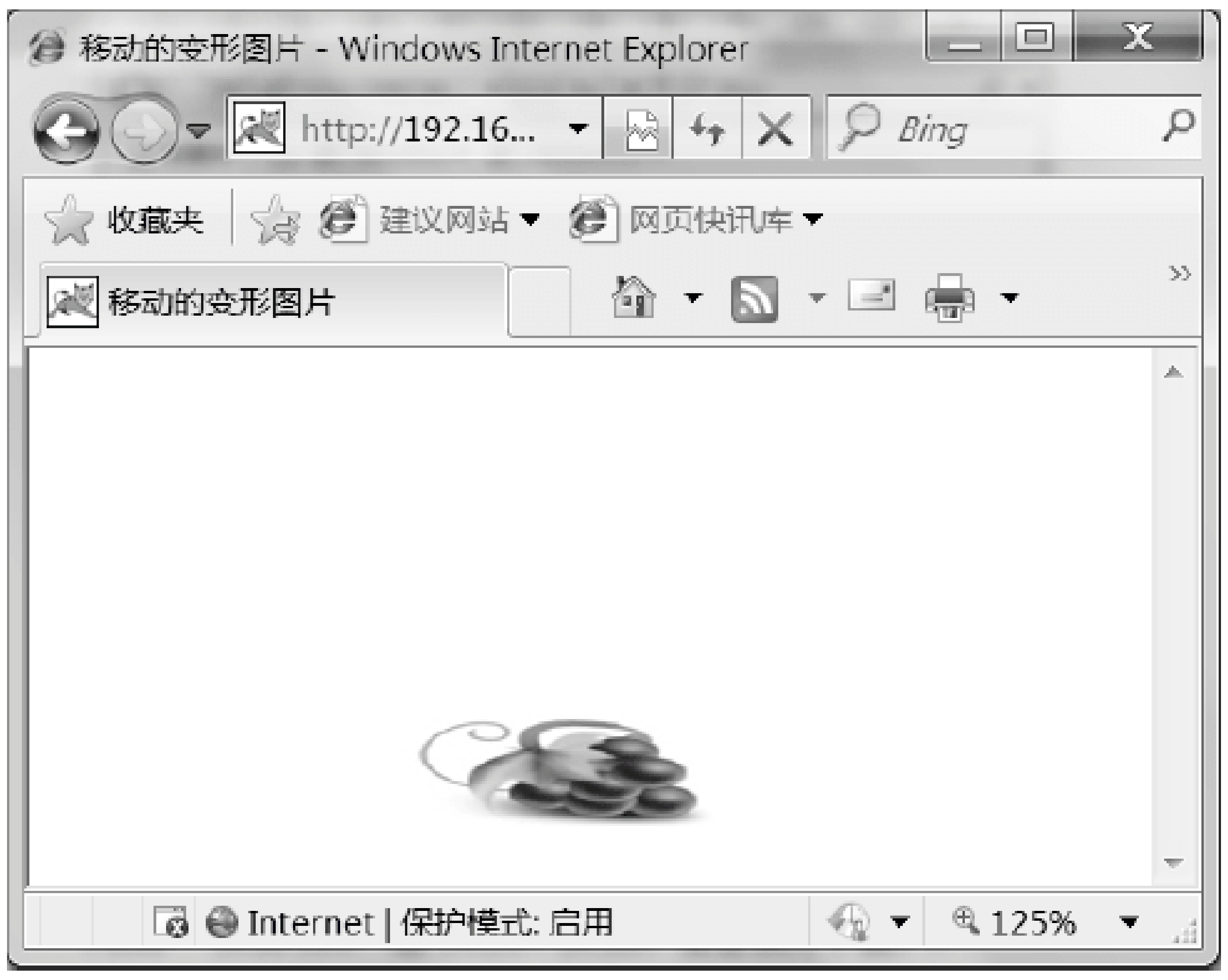


图 14.25　页面刚被载入时

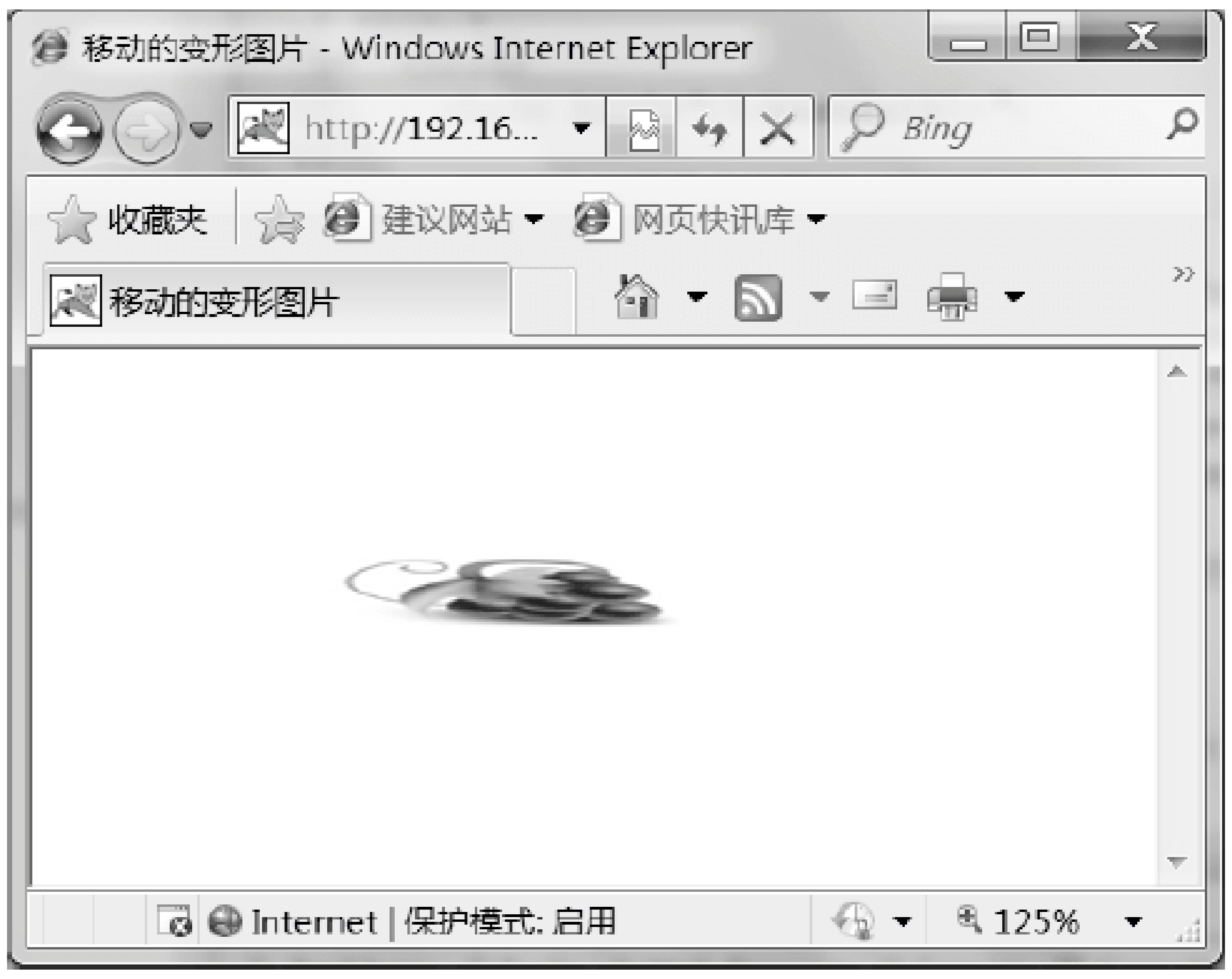


图 14.26　载入一定时间时

关键技术

本实例主要应用了 div 层和 Math 对象的 random()方法。而实现图片在移动飘动过程中变幻形状，主要是通过简单的判断使图片在指定范围变大或缩小。

本实例中，在页面被加载时，首先层会向右下角位置开始移动并且改变图片形状，当层的位置等于页面的边界时，根据 random()函数随机获取反弹后的方向，使层的 x 与 y 坐标加上指定的值或减少指定的值，然后继续运行。

设计过程

（1）利用 JavaScript 脚本实现移动的变幻形状效果的图片，代码如下：

```
<SCRIPT LANGUAGE="JavaScript">
//使图片开始随意移动
var isNS = ((navigator.appName == "Netscape") && (parseInt(navigator.appVersion) >= 4));
var _all = '';
var _style = '';
var wwidth, wheight;
var ydir = '++';
var xdir = '++';
var id1, id2, id3;
var x = 1;
var y = 1;
var x1, y1;
if(!isNS) {
        _all='all.';
        _style='.style';
}
function windowsize() {
        clearTimeout(id1);
        clearTimeout(id2);
        clearTimeout(id3);
        wwidth = document.body.clientWidth - 55;
        wheight = document.body.clientHeight - 50;

        id3 = setTimeout('randomdir()', 20000);
        animate();
}
function randomdir() {
 if (Math.floor(Math.random()*2)) {
        (Math.floor(Math.random()*2)) ? xdir='--': xdir='++';
 }
 else {
        (Math.floor(Math.random()*2)) ? ydir='--': ydir='++';
 }
 id2 = setTimeout('randomdir()', 20000);
}
```

```
function animate() {
 eval('x'+xdir);
 eval('y'+ydir);
 if (isNS)      {
         picture_div.moveTo((x+pageXOffset),(y+pageYOffset))
 }
 else    {
         picture_div.pixelLeft = x+document.body.scrollLeft;
         picture_div.pixelTop = y+document.body.scrollTop;
 }
 if (isNS)      {
         if (picture_div.top <= 5+pageYOffset) ydir = '++';
         if (picture_div.top >= wheight+pageYOffset) ydir = '--';
         if (picture_div.left >= wwidth+pageXOffset) xdir = '--';
         if (picture_div.left <= 5+pageXOffset) xdir = '++';
 }
 else    {
         if (picture_div.pixelTop <= 5+document.body.scrollTop) ydir = '++';
         if (picture_div.pixelTop >= wheight+document.body.scrollTop) ydir = '--';
         if (picture_div.pixelLeft >= wwidth+document.body.scrollLeft) xdir = '--';
         if (picture_div.pixelLeft <= 5+document.body.scrollLeft) xdir = '++';
 }
 id1 = setTimeout('animate()', 30);
}
```

（2）图片反复地变幻形状由大变小，再由小变大，代码如下：

```
var size = 1;
var bool = true;
function distortion(){
 if (document.all)
 if (bool == true){
         size++;
 }
 if(size==100) {
         size--;
         bool=false;
 }
 if(size==10){
         size++;
         bool=true;
 }
 if(bool==false){
         size--;
 }
 image1.width = 130+size;
 image1.height = 120-size;
 setTimeout("distortion()",40);
}
</script>
```

（3）在页面被加载时调用 windowsize()方法和 distortion()方法，当页面改变大小时调用 windowsize()方法，代码如下：

```
<div id="picture_div" style="position:absolute; visibility:visible; left:0px; top:0px; z-index:-1">
   <a href = "http://www.mingrisoft.com">
      <img name="image1" src="mrfish.jpg" border="0">
   </a>
</div>
<script language="javascript">
   var picture_div=eval('document.'+_all+'picture_div'+_style);
</script>
```

秘笈心法

在 JavaScript 中，navigator 表示浏览器对象，然后调用其 name 属性返回浏览器的名称。在基于 Netscape 的浏览器中，这个属性的值是 Netscape。在 IE 中，这个属性的值是 Microsoft Internet Explorer。其他浏览器可以正确地表示自己或者伪装成其他的浏览器以达到兼容性。

14.4　图片的动画效果

实例 395	图片翻转效果 光盘位置：光盘\MR\14\395	初级 实用指数：★★★

实例说明

本实例主要使用了 CSS 样式中滤镜的 fliph 和 flipv 样式来实现图片的水平翻转和垂直翻转效果。当用户单击“水平翻转”按钮时，图片会进行水平翻转；当单击“垂直翻转”按钮时，图片会进行垂直翻转，运行效果如图 14.27 和图 14.28 所示。



图 14.27　水平翻转效果的图片



图 14.28　垂直翻转效果的图片

关键技术

Flip 是 CSS 滤镜的翻转属性，FlipH 代表水平方向翻转，FlipV 代表垂直方向翻转，它们的原理都是制作一个镜像，然后让指定元素在水平或垂直方向实现翻转。

设计过程

（1）利用 JavaScript 的 image 对象实现图片的 fliphturn()水平翻转函数和 flipvturn()垂直翻转函数，代码如下：

```
<script language="javascript">
function fliphturn(){//水平翻转
	image11.style.filter = image11.style.filter =="fliph"?"":"fliph";
}
function flipvturn(){//垂直翻转
	image22.style.filter = image22.style.filter =="flipV"?"":"flipV";
}
</script>
```

（2）在网页中添加表单及相关表单元素，并使用“水平翻转”按钮调用 fliphturn()函数，使用“垂直翻转”按钮调用 flipvturn()函数，代码如下：

```
<table width="433" border="2" align="center" cellpadding="3" cellspacing="4" bordercolor="#c0c0c0" bgcolor="#FFFFFF">
 <tr>
   <td colspan="2"><div align="center"><font color="#6954ed" size="6">水平翻转</font></div></td>
  </tr>
  <tr>
    <td width="165"><font color="lightblue" face="黑体" size="4">原图：</font></td>
    <td width="252"><font color="lightblue" face="黑体" size="4">水平翻转执行结果：</font></td>
  </tr>
  <tr>
    <td><img src="test.jpg" border="0" id="image1"></td>
```

```
    <td><img src="test.jpg" border="0" id="image11"></td>
  </tr>
  <tr>
    <td colspan="2"><div align="center"><font color="#6954ed" size="6">垂直翻转</font></div></td>
  </tr>
  <tr>
    <td><font color="lightblue" face="黑体" size="4">原图：</font></td>
    <td><font color="lightblue" face="黑体" size="4">垂直翻转执行结果：</font></td>
  </tr>
  <tr>
    <td><img src="test.jpg" border="0" id="image2"></td>
    <td><img src="test.jpg" border="0" id="image22"></td>
  </tr>
</table>
<p>
<center>
<input type="button" name="button1" value="水平翻转" onClick="flipturn()">
     
<input type="button" name="button2" value="垂直翻转" onClick="flipvturn()">
</center>
```

秘笈心法

FlipH 是一个水平翻转对象的滤镜，当把 FlipH 加载到一个对象上时，该对象将产生一个水平镜像，以此来实现水平翻转的效果；FlipV 是一个垂直翻转对象的滤镜，当把 FlipH 加载到一个对象上时，该对象将产生一个垂直镜像，以此来实现垂直翻转的效果。这两个滤镜都没有参数，可直接使用。

实例 396	图片的水波倒影效果 光盘位置：光盘\MR\14\396	初级 实用指数：★★★

实例说明

图片的水波倒影的显示效果会让人感到很新颖，如有些博客网站中有人将自己的照片设置为水波倒影效果。运行本实例，在成功加载页面后，页面中会显示出水波倒影效果的图片，如图 14.29 所示。

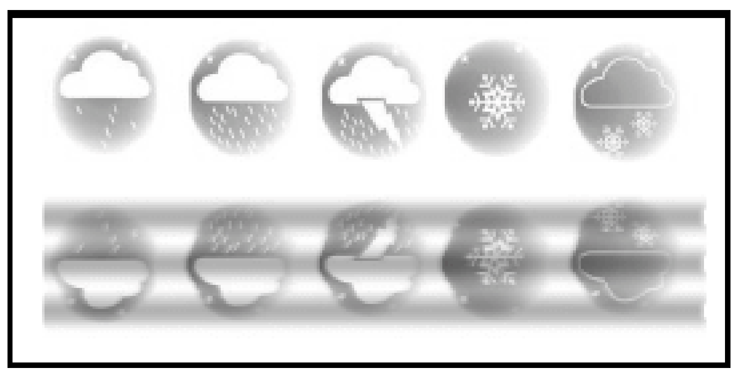


图 14.29　水波倒影效果

关键技术

本实例主要应用 CSS 样式中的 wave 滤镜来实现，语法格式如下：

```
Filter:wave(Add=true(false)，Frep=频率，LightStrength=增强光效，phase=偏移量，Strength=强度)
```

参数说明

❶ Add：表示对象是否按照波纹样式打乱，取值为 true 或 false。

❷ Freq：表示生成波纹的频率，也就是指定在对象上共需要多少个波纹。

❸ LightStrength：表示生成波纹增强光效，取值范围在 0～100 之间。

❹ phase：用于设置正弦波开始的偏移量，这个值的通用值为 0，取值范围在 0～100 之间。这个值代表开始时偏移量占波长的百分比，如值为 25 时代表正弦波从 90 度（360*25%）的方向开始。

设计过程

利用 JavaScript 实现水波倒影效果，关键代码如下：

```
<script language="JavaScript">
function map()      {
        setInterval("img1.filters.wave.phase+=10",100);
}
if (document.all)    {
        document.write('<Center><img  id=img1  src="'+document.all.mappedimg.src+'"  style="filter:wave(strength=3,freq=3,phase=0,lightstrength=30)
blur() flipv()"></Center>');
```

```
        window.onload = map;
}
</script>
```

秘笈心法

本实例首先是在页面中添加了一幅图片，然后通过自定义 JavaScript 函数 map()动态创建一幅图片，最后在自定义函数中将动态创建的图片的滤镜设置为水波倒影（wave）特效。

实例 397	图片渐隐渐现 光盘位置：光盘\MR\14\397	初级 实用指数：★★★

实例说明

在美化页面时，有一种渐隐渐现的图片效果会给人一种好奇心理。本实例是把图片从模糊渐渐变换成清晰状态，再由清晰渐渐变为模糊状态，反复执行此操作，最后实现图片渐隐渐现的效果，运行结果如图 14.30 和图 14.31 所示。



图 14.30　渐变为清晰状态的图片

图 14.31　渐变为模糊状态的图片

关键技术

本实例主要应用了 CSS 样式中的滤镜 alpha 的 opacity 属性来改变图片的透明度，并且通过一些简单的条件判断，使图片能够在指定的时间内循环渐隐渐现。

设计过程

（1）利用 JavaScript 实现渐隐渐现的效果，代码如下：

```
<script language="JavaScript">
var b = 1;
var c = true;
function show(){
   if(document.all)
   if(c == true)
         b++;
   else
         b--;
   if(b==100){
      b--;
      c = false
   }
   if(b==10) {
      b++;
      c = true;
   }
   myImage.filters.alpha.opacity=b;
   setTimeout("show()",25);
}
show();
</script>
```

（2）在页面中添加一个图片使用滤镜效果，代码如下：

```
<img src="dh.jpg" name="myImage" border="1" style="filter:alpha(opacity=0)">
```

秘笈心法

在 JavaScript 中，document 对象的 all 属性是 IE 4.0 及以上版本的专有属性，是一个表示当前文档的所有对象的数组，它不仅包括页面上可见的实体对象，还包括一些不可见的对象，如注释。在 all 数组中元素不分层次，它是按照在文档中出现的先后顺序平行地罗列。

实例 398 图片的探照灯效果

光盘位置：光盘\MR\14\398

初级

实用指数：★★★

实例说明

本实例将使用 JavaScript 编写一个图片探照灯的特效，当页面被加载时，这个探照灯的效果会在图上重复左右扫描，将其中一部分显示为光亮状态而其他部分显示为灰暗状态，实例运行结果如图 14.32 和图 14.33 所示。



图 14.32　探照灯效果 1



图 14.33　探照灯效果 2

关键技术

本实例使用了 CSS 滤镜技术中的 light 属性，通过该属性调用 addPoint()和 MoveLight()方法来设置图片上光源的大小，并移动光源。

设计过程

（1）使用 JavaScript 实现探照灯效果，代码如下：

```
<script language="javascript">
if (document.all && window.imagelight){
var x=new Array();
var heading=new Array();
var y=new Array();
if (imagelight.length==null){
        imagelight[0]=document.all.imagelight;
        x[0]=0;
        heading[0]="right";
        y[0]=imagelight[0].height;
        imagelight[0].filters.light.addPoint(100,50,100,255,255,255,90);
}
else
        for (i=0;i<imagelight.length;i++){
        x[i]=0;
        heading[i]="right";
        y[i]=imagelight[i].height;
        imagelight[i].filters.light.addPoint(100,50,100,255,255,255,90);
}
}
function light(cur){
imagelight[cur].filters.light.MoveLight(0,x[cur],y[cur],200,-1);
if (x[cur]<imagelight[cur].width+200&&heading[cur]=="right")
        x[cur]+=10;
else if (x[cur]>imagelight[cur].width+200){
        heading[cur]="left";
        x[cur]-=10;
}
else if (x[cur]>-200&&x[cur]<-185){
```

```
        heading[cur]="right";
        x[cur]+=10;
    }
    else{
        x[cur]-=10;
        heading[cur]="left";
    }
}
if (document.all&&window.imagelight){
 if (imagelight.length==null)
        setInterval("light(0)",imagelight[0].speed);
 else
        for (t=0;t<imagelight.length;t++){
        var temp='setInterval("light('+t+')",'+imagelight[t].speed+')';
        eval(temp);
        }
}
</SCRIPT>
```

（2）在页面中编写 CSS 样式，并标记一幅图片，代码如下：

```
<STYLE type=text/css>
#imagelight {
 FILTER: light
}
</STYLE>
<center>
<IMG id="imagelight" src="test.jpg" speed="20">
</center>
```

秘笈心法

本实例使用了 CSS 样式中的 addPoint()方法，它是为图片上的光源滤镜增加点光的，而 movelight()方法是用于移动锥形光的焦点，也就是图片上光源的焦点或光源的原点，通过这两个方法的 x、y 属性来达到控制光源向左或向右移动的效果。

实例 399	雷达扫描式图片效果 光盘位置：光盘\MR\14\399	初级 实用指数：★★★

实例说明

在高科技设备领域里都有雷达扫描装置，当雷达运作时，就会有一个 45%的图片在雷达中旋转。本实例将使用 JavaScript 实现类似雷达扫描的图片效果，运行结果如图 14.34 所示。

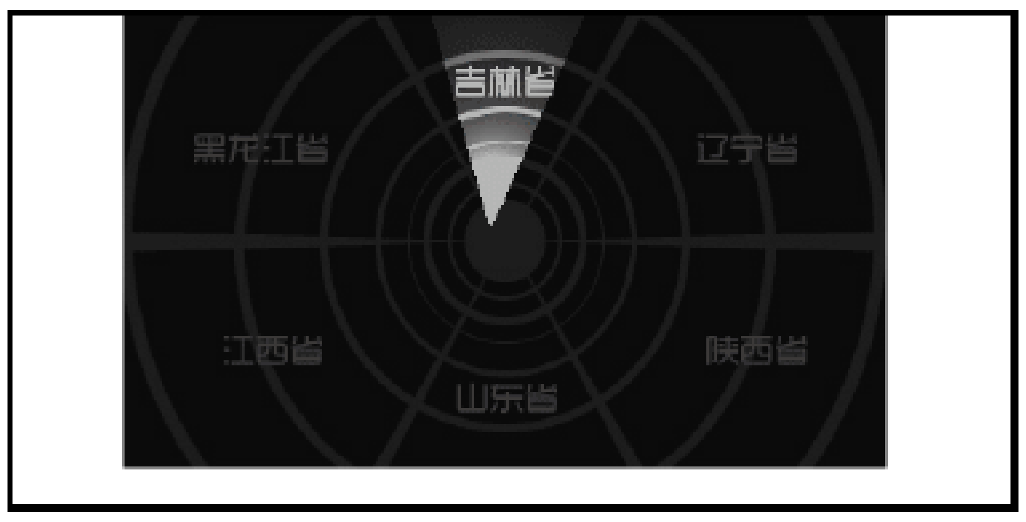


图 14.34　雷达扫描效果的图片

关键技术

本实例主要使用 Math 对象的 cos()和 sin()方法来获取某一个角度，然后使图片根据此角度进行旋转。本实例通过 CSS 样式的图片滤镜的 3 个方法，来实现雷达扫描特效的方式。

- moveLight：移动图片的照明位置。
- light.addCone：设置图片的照明中心点及照明度的大小。
- light.addAmbient：设置图片的阴影效果。

设计过程

（1）利用 JavaScript 编写雷达扫描图片的效果，代码如下：

```
<script language="javascript">
   var X=20//角度
```

```
    var Y=20//角度
    var Z=40//角度
    var xInc=100;
    var yInc=100;
    var r=200;
    var X1=0;
    var change = (2 * Math.PI)/360
function movearea(){
        X = r+r*Math.cos(X1*change);
        Y = r+r*Math.sin(X1*change);
        X1 += 10;
        if(X1 == 360)    X1=0;
                img1.filters[0].moveLight(0,X,Y,Z,1);
                mytimeout=setTimeout('movearea()',80);
        }
function beginscan(){
        img1.filters.light.addCone(192,110,0,X,Y,0,255,0,150,20);
        img1.filters.light.addAmbient(0,255,0,50)
        var x=0;
        movearea();
}
</script>
```

（2）在页面中添加一幅图片，并在页面被加载时调用 beginscan()函数，代码如下：

```
<title>雷达式扫描图片特效</title>
</head>
<body onLoad="beginscan();">
<center>
<IMG SRC="test.jpg" ID="img1" STYLE="filter:light()">
</center>
```

秘笈心法

本实例在设置雷达扫描效果的光源时，用到了 CSS 样式中的 addAmbient()方法，用于设置包围的光源，语法格式如下：

```
addAmbient (iRed,iBlue,iStrength)
```

参数说明

❶ iRed：必选参数，整数值，指定红色值，取值范围在 0～255 之间。

❷ iBlue：必选参数，整数值，指定蓝色值，取值范围在 0～255 之间。

❸ iStrength：必选参数，整数值，指定光源强度，取值范围在 0～100 之间。

其实本例的效果就是滤镜添加环境光，环境光是无方向的，并且均匀地洒在页面的表面，环境光有颜色和强度值，可以为对象添加更多的颜色，它通常和光源一起使用，无返回值。

实例 400	在页面中旋转的图片效果 光盘位置：光盘\MR\14\400	初级 实用指数：★★★

实例说明

在一些交易平台网站，如淘宝网、拍拍网等，经常会发现一些图片以圈的形状旋转的效果，如此反复地执行。本实例将实现图片的旋转效果，运行结果如图 14.35 所示。

关键技术

本实例主要应用了 Math 对象的 sin()和 cos()方法，通过取得正弦值和余弦值，然后加一些算法来改变当前层的位置，使图片在页面中旋转。

设计过程

（1）利用 JavaScript 的 Math 对象的正弦和余弦值来改变当前层的位置，代码如下：

```
<script language="javascript">
var x1=200;
var x2=200;
var timer;
var r=60;
var i = 0;
function eddyphoto(i) {
	var ob=document.all("divround");
	ob.style.posTop = r*Math.sin((i*Math.PI)/180)+x1;
	ob.style.posLeft = r*Math.cos((i*Math.PI)/180)+x2;
	i=i+1;
	if (r>100){
		window.clearTimeout(timer);
	}
	else{
		if (i > 360){
			i = 0;r = r + 1;
		}
		timer=setTimeout("eddyphoto("+i+")",10);
	}
}
eddyphoto(0);
</script>
```

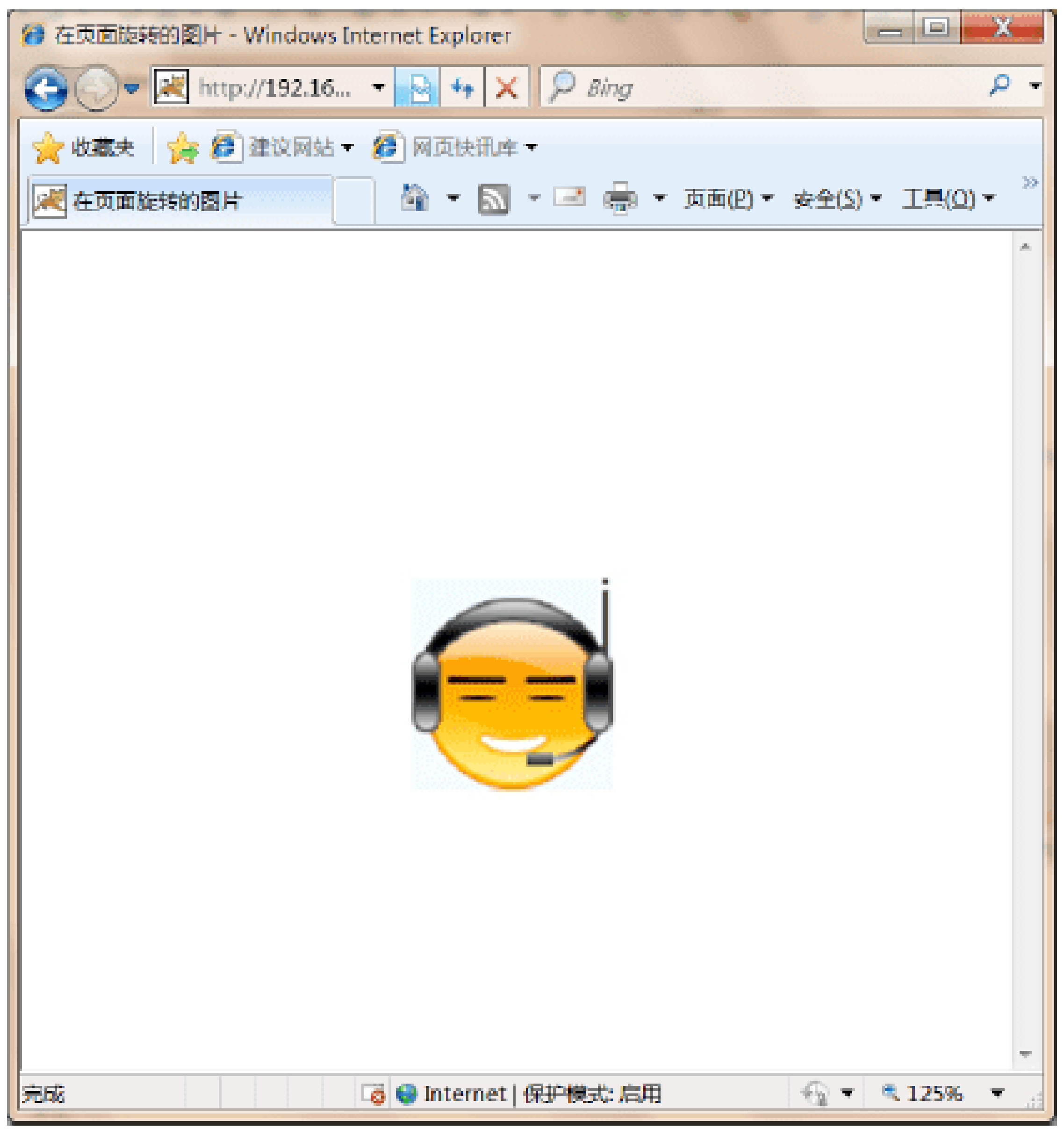


图 14.35　页面旋转的图片

（2）在页面中添加一个层，并在层中添加要旋转效果的图片，代码如下：

```
<div id="divround" style="width:50pt; top:198.75pt; left:256.5pt; position:absolute; z-index:0">
<img src="temp.jpg">
</div>
```

秘笈心法

本实例在获取<div>层的顶端和左边时，用到了 posTop 和 posLeft 属性，其中 posTop、posLeft 和 Top、Left 一样，但区别在于，Top 和 Left 固定了元素单位为 px，而 PosTop、PosLeft 只是一个数值。

实例 401　改变形状的图片

初级

光盘位置：光盘\MR\14\401

实用指数：★★★

实例说明

本实例将实现一个改变形状的图片。运行本实例，在页面中的居中位置将显示一张图片，之后图片会开始改变形状，先由大变小，当图片达到指定的大小时，图片又会由小变大，运行结果如图 14.36 所示。



图 14.36　改变形状的图片

关键技术

本实例的实现很简单，主要通过一些简单的 if 条件语句的判断，动态修改图片的宽度和高度，使图片能够在指定的范围内变大变小。

设计过程

（1）创建用于实现改变图片形状大小的 JavaScript 函数，关键代码如下：

```
<script language="javascript">
//改变图片形状
var size = 1;
var bool = true;
function imgsize(){
 if (document.all)
 if (bool == true){
        size++;
 }
 if(size==100) {
        size--;
        bool=false;
 }
 if(size==10)  {
        size++;
        bool=true;
 }
 if(bool==false){
        size--;
 }
 image1.width = 130 + size;
 image1.height = 120 - size;
 setTimeout("imgsize()",30);
}
</script>
```

（2）在页面中添加图片，并在页面加载时调用 imgsize()函数，代码如下：

```
<body onLoad="imgsize()">
<center>
 <img name="image1" src="mr.jpg" border="0">
</center>
```

秘笈心法

本实例主要是在自定义的 JavaScript 函数中，首先获取图片对象，然后再设定判断范围，最后通过 setTimeout()函数循环执行这个自定义的 JavaScript 函数，在函数中动态修改图片的宽度和高度，从而实现预期效果。

14.5　选择头像图片

头像选择在网络程序中的应用十分频繁，它不仅使用很方便，而且可以使整个网页看起来更有活力。例如，在论坛中需要选择自己喜欢的头像来模拟个人的肖像，这种接近现实生活的模拟既形象又生动，使网页增色不少。

实例 402 | **在列表中选择图片头像** | 初级
光盘位置：光盘\MR\14\402 | 实用指数：★★★

实例说明

在论坛或留言簿的用户注册页面中，加入头像的选择功能，可以使网站增色不少。运行本实例，在“头像选择”下拉列表框中选择某个数字时，将在下面的列表框中显示该数字所代表的头像，如图 14.37 所示。



图 14.37　选择头像图片

关键技术

本实例主要通过下拉列表框的 onChange 事件调用 JavaScript 自定义函数实现，在该自定义函数中，关键是应用下拉列表框的 selectedIndex 属性和 option[n].value 属性，以此来获取选择头像所对应的文件名，并赋值给 Image 对象的 src 属性，用于在页面中实时显示头像信息。

设计过程

（1）创建 index.jsp 页面，并在该页面中添加下拉列表，选项的值为图片文件的名称，下拉列表框的 onChange 事件执行的操作是调用 showimg()函数，代码如下：

```
<tr>
      <td>头像选择：</td>
      <td>
      <select name="selectimage" size="1" id="selectimage" onChange="showimg();">
            <option selected>1</option>
            <option>2</option>
            <option>3</option>
            <option>4</option>
            <option>5</option>
            <option>6</option>
            <option>7</option>
            <option>8</option>
            <option>9</option>
            <option>10</option>
            <option>11</option>
            <option>12</option>
            <option>13</option>
            <option>14</option>
            </select></td>
</tr>
<tr>
      <td> </td>
      <td ><img src="Images/1.jpg" name="img" width="60" height="60" border="1"></td>
</tr>
```

（2）自定义 JavaScript 函数 showimg()，用于实现用户改变下拉列表的值时显示相应的图片信息，代码如下：

```
<script language="javascript">
function showimg()
{
 document.form1.img.src = "Images/"+document.form1.selectimage.selectedIndex+".jpg";
}
</script>
```

秘笈心法

通过本实例，读者可以实现在 QQ 聊天室中通过下拉列表选择头像的功能，还可以实现在人事档案管理系统中通过下拉列表选择头像来选择员工的个人肖像。

实例 403 在弹出的新窗口中选择图片

光盘位置：光盘\MR\14\403

初级

实用指数：★★★

实例说明

在用户注册的网页中加入头像选择功能可以使网站更具有吸引力，而在选择头像时通过弹出的网页对话框选择，则更直观、方便。运行本实例，单击图 14.38 中的“更改头像”超链接，将弹出如图 14.39 所示的网页对话框，在用户选择某个头像后，当前窗口将自动关闭，并且超链接上将会显示刚刚选择的头像，如果在头像选择页面没有任何选择动作，则将显示默认的头像。



图 14.38 更改头像



图 14.39 在页面中选择头像

关键技术

本实例主要是通过 window 对象的 showModalDiaglog()方法实现的，但需要注意的是，如果用户在此页面中没有找到想要的头像，且直接关闭了对话框，若此处不做处理，则页面中显示图片的位置将会显示一个无图片的框。为了解决这个问题，需要使用 JavaScript 的 Global 对象的 undefined 属性，语法格式如下：

```
undefined
```

功能：返回 undefined 的一个初始值。

说明： undefined 属性是 Global 对象的一个成员，此属性在脚本引擎初始化后可用，如果已声明了一个变量但还没有初始化，那么该变量的值就是 undefined；如果还没有声明变量，那么就不能将其与 undefined 进行比较，但是可以将该变量的类型与字符串 undefined 进行比较。

设计过程

（1）在页面中添加默认头像和“更改头像”超链接，此链接触发的是执行 JavaScript 自定义函数 deal()，

关键代码如下：

```
<table width="18%" height="125"  border="1" align="center" cellpadding="3" cellspacing="2" bordercolor="#0066FF" bgcolor="#99ccff">
  <tr>
    <td height="89"><div align="center"><img src="Images/1.jpg" name="img" width="60" height="60"></div></td>
  </tr>
  <tr>
    <td height="30"><div align="center"><a href="#" onClick="deal();">[更改头像]</a></div></td>
  </tr>
</table>
```

（2）JavaScript 自定义 deal()函数，是在打开网页对话框时，首先会自己选择头像，如果选中并单击头像就把值赋给“更改头像”图片的 src 属性，代码如下：

```
<script language="javascript">
function deal(){
        var someValue;
        someValue=window.showModalDialog('ImageList.jsp','','dialogWidth=520px;\
        dialogHeight=400px;status=no;help=no;scrollbars=no;resizable=yes');
        if (someValue == undefined){
                someValue="0";
        }
        document.img.src="Images/"+someValue+".jpg";
}
</script>
```

（3）创建 ImageList.jsp 页面用于存放头像列表，此页显示全部的头像信息，并提供返回更改头像页面的超链接，主要代码如下：

```
<tr align="center">
<td valign="top"><a href="#" onClick="selectImage('1')">
<img src="Images/1.jpg" width="60" height="60" border="0"></a></td>
        <td valign="top"><a href="#" onClick="selectImage('2')">
<img src="Images/2.jpg" width="60" height="60" border="0"></a></td>
          <td valign="top"><a href="#" onClick="selectImage('3')">
<img src="Images/3.jpg" width="60" height="60" border="0"></a></td>
        <td valign="top"><a href="#" onClick="selectImage('4')">
<img src="Images/4.jpg" width="60" height="60" border="0"></a></td>
          <td valign="top"><a href="#" onClick="selectImage('5')">
<img src="Images/5.jpg" width="60" height="60" border="0"></a></td>
  </tr><tr align="center">
    <td valign="top"><div align="center" class="style1">头像 1</div></td>
      <td valign="top"><div align="center" class="style1">头像 2</div></td>
        <td valign="top"><div align="center" class="style1">头像 3</div></td>
        <td valign="top"><div align="center" class="style1">头像 4</div></td>
        <td valign="top"><div align="center" class="style1">头像 5</div></td>
  </tr>
```

秘笈心法

应用 Global 对象的 undefined 属性，当显式地测试变量或将变量设置为 undefined 时，undefined 是很有用的。

14.6　图片的其他效果

实例 404	页面中播放图片 光盘位置：光盘\MR\14\404	初级 实用指数：★★★

实例说明

本实例将实现一个类似幻灯片播放图片的效果。运行本实例，单击“下一张”按钮，会显示下一张图片，单击“上一张”按钮，则会显示上一张图片。如果用户 3 秒内没有任何操作，图片会自动播放至下一张，运行结果如图 14.40 所示。



图 14.40　幻灯片播放图片

关键技术

本实例主要应用了 CSS 样式的 filter 滤镜的 revealTrans 属性，该属性提供了更加多变的转换效果。下面对 revealTrans 属性进行详细说明。

revealTrans 属性的语法格式如下：

```
revealTrans(Transition,Duration)
```

参数说明

❶ Transition：表示切换方式，它有 24 种方式。

❷ Duration：表示切换时间，以秒为单位。

revealTrans 属性的 Transition 参数取值如表 14.3 所示。

表 14.3　revealTrans 属性的 Transition 参数取值

Transition 参数值	切 换 效 果	Transition 参数值	切 换 效 果
0	矩形从大至小	12	随机溶解
1	矩形从小至大	13	从上下向中间展开
2	圆形从大至小	14	从中间向上下展开
3	圆形从小至大	15	从两边向中间展开
4	向上推开	16	从中间向两边展开
5	向下推开	17	从右上向左下展开
6	向右推开	18	从右下向左上展开
7	向左推开	19	从左上向右下展开
8	垂直形百叶窗	20	从左下向右上展开
9	水平形百叶窗	21	随机水平细纹
10	水平棋盘	22	随机垂直细纹
11	垂直棋盘	23	随机选取一种特效

设计过程

（1）首先自定义翻动图片，代码如下：

```
<!--定义翻动图片的自定义函数-->
<script language="javascript">
var plPic = new Image();
var gIndex = 0;
function SlideImg(index){
	gIndex = index;
	if ('Microsoft Internet Explorer' == navigator.appName){
		document.images["SlideImg"].filters.item(0).Apply();
	}
	document.images["SlideImg"].src = sPicArr[index][0];
```

```
        if ('Microsoft Internet Explorer' == navigator.appName){
                document.images["SlideImg"].filters.item(0).play();
        }
```

（2）利用 JavaScript 自定义函数 NextImg()和 PrevImg()，代码如下：

```
<!--上一张、下一张-->
function PrevImg(){
        gIndex = ((gIndex-1)<0?(sPicArr.length-1):(gIndex-1));
        SlideImg(gIndex)
}
function NextImg(){
        gIndex = ((gIndex+1)>=sPicArr.length?0:(gIndex+1));
        SlideImg(gIndex);
}
```

（3）编写自动播放的自定义函数 autoImg()，代码如下：

```
<!--自动播放-->
var sid;
function autoImg(){
        if(sid==null)
                sid = setInterval('NextImg()', 3000);
}
```

（4）创建图片集合并设置其路径，代码如下：

```
<!--设置图片的路径-->
<script language="javascript">
var sPicArr = new Array();
sPicArr[0] = new Array("images/1.jpg");
sPicArr[1] = new Array("images/2.jpg");
sPicArr[2] = new Array("images/3.jpg");
sPicArr[3] = new Array("images/4.jpg");
sPicArr[4] = new Array("images/5.jpg");
sPicArr[5] = new Array("images/6.jpg");
sPicArr[6] = new Array("images/7.jpg");
sPicArr[7] = new Array("images/8.jpg");
sPicArr[8] = new Array("images/9.jpg");
sPicArr[9] = new Array("images/10.jpg");
sPicArr[11] = new Array("images/11.jpg");
sPicArr[12] = new Array("images/12.jpg");
sPicArr[13] = new Array("images/13.jpg");
sPicArr[14] = new Array("images/14.jpg");
</script>
```

（5）在页面中调用自定义的函数和图片，代码如下：

```
<body onLoad="autoImg();">
<table border="1" align="center" cellpadding="3" cellspacing="2" bordercolor="lightblue">
  <tr>
    <td align="center"><input name="firstbutton" type="button" id="firstbutton" value="上一张" onClick="PrevImg();">
    <input name="nextbutton" type="button" id="nextbutton" value="下一张" onClick="NextImg();"></td>
    <td >
<div align="center"><img src="images/1.jpg" name="SlideImg" width="60" height="60" border="1" id="SlideImg" style="BORDER-TOP: #000 1px solid; FILTER: revealtrans(duration=2.0,transition=5); BORDER-BOTTOM: #000 1px solid" onMouseOut=;>
</div></td>
  </tr>
</table>
</body>
```

秘笈心法

根据本实例的实现，读者可以实现在网站中广告图片的播放，还可以实现在网站中新网栏目的新闻图片的播放。

实例 405	导航地图 光盘位置：光盘\MR\14\405	初级 实用指数：★★★

实例说明

在开发网络程序时，通常需要对网站的功能流程进行详细说明。程序的开发者很重视如何将某个网站的功

能清晰地描述出来。解决这一问题的最好办法就是制作导航地图，该地图不仅可以美化网站，同时也可以更加详细地说明该网站的具体功能及操作流程，实例运行结果如图 14.41 所示。

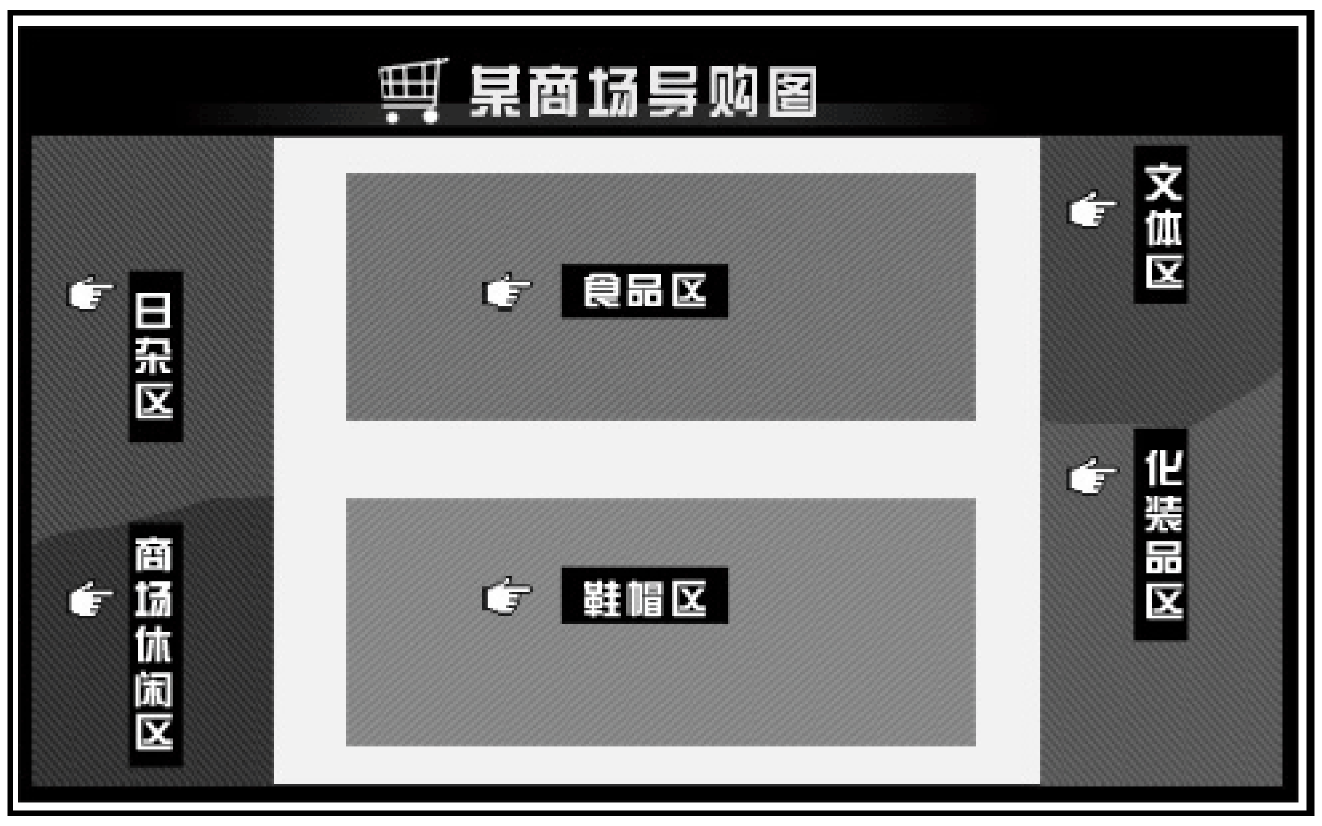


图 14.41　导航地图

关键技术

本实例主要通过建立图像地图实现导航地图。图像地图通常被用在设置图像链接或图像导航方面。可以通过设置图像地图热区的链接，完成商场内各个商品专区的链接。设置图像地图的“替代”文本可以在图像中起到对商品专区导航的作用。建立图像热区也就是建立图像地图。图像地图“属性”面板如图 14.42 所示。

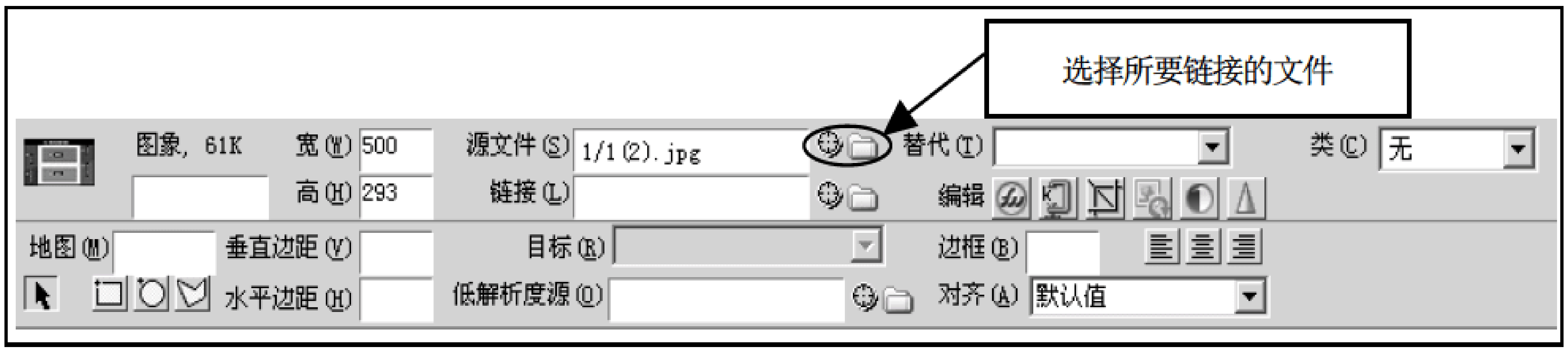


图 14.42　图像地图“属性”面板

图像“属性”面板中的图像地图工具共分为 3 种，分别是矩形、圆形和多边形。建立图像地图非常简单，只要选择任意一种图形后，在适当位置按住左键拖动鼠标即可建立相应热区。

通过单击图像热区“属性”面板中“链接”域后的浏览图像文件夹图标添加相应图标，在弹出的“选择文件”对话框中选择所要链接的文件，或者直接输入超链接的路径及文件名。

设计过程

（1）首先在 Photoshop 中制作需要建立图片热区的图片。

（2）在 Dreamweaver 中选择“插入”→“图像”命令，插入一张图片，使用鼠标左键单击选中此图片，此时在“属性”控制面板中可以看到此图片的相关属性。

（3）初次使用“属性”控制面板时，其默认为“简化”形式，单击右下方的向下箭头可打开其扩展属性模式。

（4）此时可以通过“属性”控制面板设置图像热区。

秘笈心法

根据本实例，读者可以在开发宠物网站程序时，对宠物进行购买时使用导航地图；在开发医药管理系统时，对药品进行出库/入库管理时使用导航地图；在开发电子商城网站时，对成功进入商城后进行商品选购时使用导航地图。

第 15 章

多媒体应用

- 播放音乐
- 插入 Flash 动画
- 播放视频

15.1 播 放 音 乐

实例 406	为网页设置背景音乐 光盘位置：光盘\MR\15\406	高级 实用指数：★★★★

实例说明

程序员在进行网络程序开发时，经常喜欢在网页中添加一些特殊的效果。例如，为网页设置背景音乐，当浏览者登录网站时，设置的背景音乐将会播放。本实例便实现了这一功能，运行结果如图 15.1 所示。

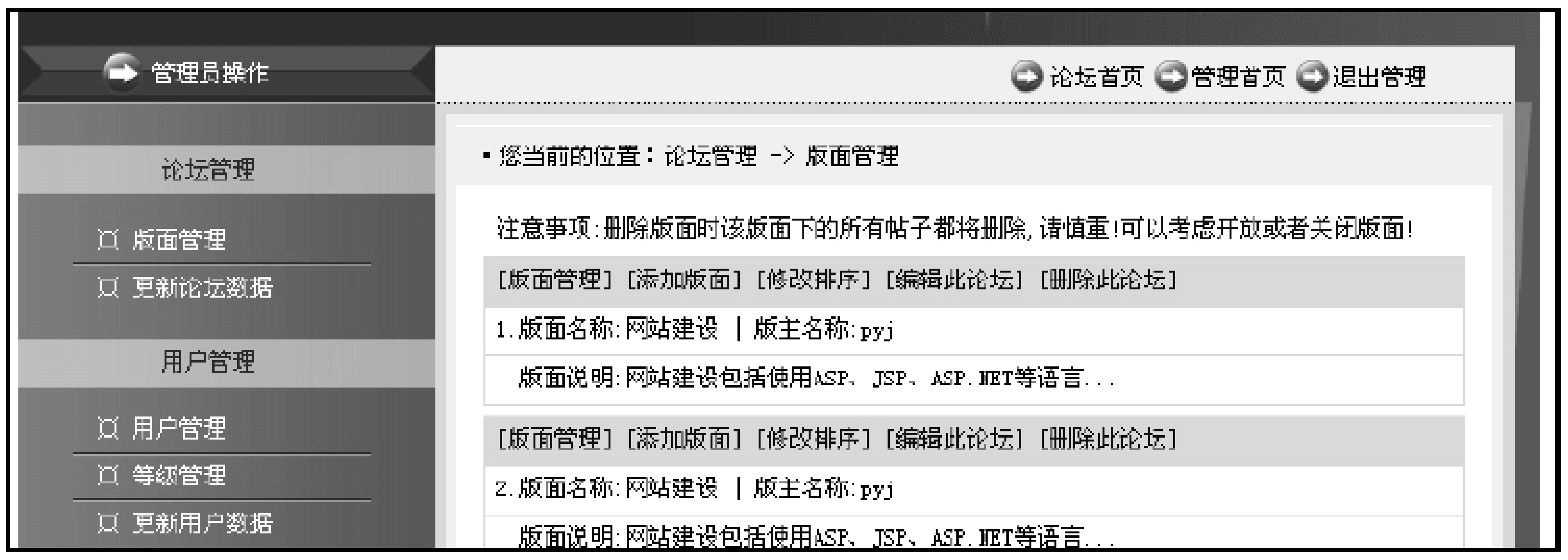


图 15.1 为网页设置背景音乐

关键技术

本实例主要应用<bgsound>标记实现为网页设置背景音乐的效果，通过该标记可以嵌入多种格式的音乐文件。<bgsound>标记的语法格式如下：

```
<bgsound src="file_name" loop="loop_name" >
```

参数说明

❶ src：背景音乐的路径。

❷ loop：播放的循环次数，取值为-1 或者 Infinite 时表示无限次循环。

注意：使用标记<bgsound>嵌入背景音乐的网页，在网页最小化时，音乐停止。

设计过程

（1）创建 index.jsp 页面，在该页面中添加指定表格和表单元素，关键代码如下：

```
<table width="775" height="741" border="0" align="center" cellpadding="0" cellspacing="0">
  <tr>
    <td width="775" height="433" background="ht_2.jpg"> </td>
  </tr>
</table>
```

（2）应用<bgsound>标记向指定的网页中添加背景音乐，代码如下：

```
<bgsound src="py.MP3" loop="-1">
```

秘笈心法

bgsound 支持的音乐格式有 wav、mid、mp3 等。在此标签中还有以下两个参数比较常用。

- volume：用于控制音量大小，取值范围在-1000～0 之间。 0 为最大音量。
- balance：表示左右声道，取值范围在-1000～1000 之间。负值为将声音发送给左声道，正值为将声音发送给右声道，0 则为立体声。

bgsound 并不是标准的标签，所以对浏览器的支持并不好，它只限于 IE，而在 Netscape 和 Firefox 中并不适用。

实例 407　随机播放背景音乐

光盘位置：光盘\MR\15\407

高级

实用指数：★★★★

实例说明

在编写网络程序时，可以为指定的网页添加背景音乐，设置的背景音乐也可以是随机的，当浏览者浏览网页时每次播放的背景音乐都是不一样的，这样可以使浏览者有一种新鲜感而不会因为循环地听着同一首音乐而感到厌倦，实例运行结果如图 15.2 所示。



图 15.2　随机播放背景音乐

关键技术

实现本实例时，首先需要应用 Array 对象将背景音乐的文件名保存到数组中，然后随机选择要播放的文件名，并通过<bgsound>标记播放背景音乐。下面对 Array 对象进行详细介绍。

Array 对象用于创建数组，数组是有序数据的集合。每一个数组中的每个元素都是一个独立的值，并且使用 new 关键字来定义数组，数组的索引值从 0 开始。Array 对象的常用属性如表 15.1 所示。

表 15.1　Array 对象的常用属性

属　　性	描　　述
index	对于一个由规则表达式匹配生成的数组，这个属性返回此匹配的索引位置
input	对于一个由规则表达式匹配生成的数组，这个属性返回原始字符串
length	返回数组中的元素个数
prototype	用于在定义数组时添加新的属性和方法，prototype 是数组对象的静态属性

Array 对象常用的方法如表 15.2 所示。

表 15.2　Array 对象常用的方法

方　　法	描　　述
concat	返回一个新数组，这个新数组是由两个或更多个数组组合而成的
join	返回字符串值，其中包含了连接到一起的数组的所有元素，元素由指定的分隔符分隔开来
toLocaleString	返回一个日期，该日期使用当前区域设置并已被转换为字符串
toString	返回对象的字符串表示
ubounds	返回在 VBArray 的指定维中所使用的最大索引值

设计过程

（1）创建 index.jsp 页面，根据需要在指定的页面中添加表格或表单元素，关键代码如下：

```
<table width="785" height="668" border="0" align="center" cellpadding="0" cellspacing="0">
  <tr>
```

```
        <td width="784" height="668" background="0.gif"> </td>
    </tr>
</table>
```

（2）应用 Array 对象实现随机背景音乐的播放，关键代码如下：

```
<script language="javascript">
var arrayname=new Array();
for (var i=0;i<5;i++)
{
        arrayname[i]="music"+(i+1)+".mp3";
}
var x=(Math.floor(Math.random()*4)+1);
document.write('<bgsound src='+'"'+arrayname[x]+'"'+' loop="infinite">');
</script>
```

秘笈心法

本实例主要应用 JavaScript 脚本，把背景音乐文件保存到 Array()数组中，然后使用 for 语句循环这个 Array 数组下的每一个元素，再应用 Math 对象的 floor()和 random()方法随机返回这个数组的一个元素，最后使用 document 对象的 write()方法输出一段<bgsound>标签的背景音乐。

实例 408　MIDI 音乐选择

光盘位置：光盘\MR\15\408

高级

实用指数：★★★★

实例说明

在进行网络程序开发时，有时需要为网站添加一些背景音乐或 MIDI 音乐等。同时也可以向网页中添加多个音乐，如何在网页中对添加的多个音乐进行选择性播放？如本实例主要对网页中的MIDI 音乐进行选择播放。运行本实例，通过下拉列表框选择所要播放的 MIDI 音乐，单击“播放 MIDI 音乐”按钮进行音乐的播放，程序运行结果如图 15.3 所示。

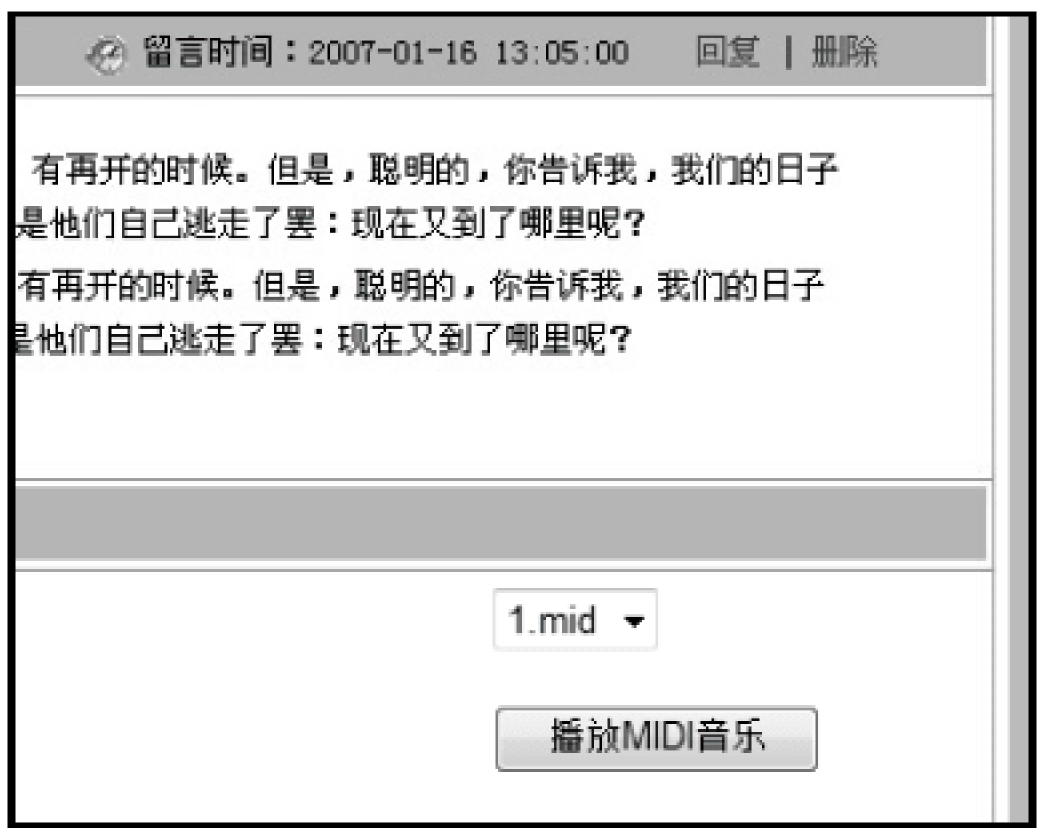


图 15.3　MIDI 音乐选择

关键技术

本实例主要应用 Options 数组实现 MIDI 音乐选择，下面将对该数组进行详细介绍。

Options 数组是 Select 对象的一个属性，即选择框中的所有选项（<OPTION>）的一个列表。Options 数组的相关属性如表 15.3 所示。

表 15.3　Options 数组的相关属性

属　性	描　述
defaultSelected	选项列表中的默认选项
index	选项列表中某选项的索引位置
length	选项列表中的选项数（<OPTIONS>）
name	选项列表的名字（NAME 特性）
selected	表示选项列表中某选项<OPTION>是否被选中的一个布尔类型值
selectedIndex	选项列表中已选中的<OPTION>的索引（位置）
text	选项列表中<OPTION>标记后的文本
value	选项列表中的 VALUE 特性

设计过程

（1）创建 index.jsp 页面，在该页面中添加所需要的表单及相关元素，关键代码如下：

```
<FORM NAME="form1">
<SELECT NAME="list" >
   <option value="1.mid">1.mid</option>
   <option value="2.mid">2.mid</option>
   <option value="3.mid">3.mid</option>
   </SELECT>
<P align="left" ><INPUT TYPE=BUTTON VALUE="播放 MIDI 音乐" onClick="Mycheck(form1.list.options[form1.list.selectedIndex].value)">
</FORM>
```

（2）通过自定义 JavaScript 脚本定义 Mycheck()函数，用于实现打开一个窗口，代码如下：

```
<SCRIPT LANGUAGE="JavaScript">
function Mycheck(sml){
        ggg=window.open(sml,"ming","toolbar=no,location=no,directories=no,status=no,scrollbars=no,resizeable=no,
        copyhistory=no,width=200,height=30")
}
</SCRIPT>
```

秘笈心法

Java 在多媒体处理方面的确优势不大，但是在程序中有时又需要一些音乐作为点缀，如果要播放音乐为 wav 等波形音频文件，此时最好的格式就是 MIDI 文件。

实例 409	在线连续播放音乐 光盘位置：光盘\MR\15\409	高级 实用指数：★★★★

实例说明

现在很多网站都提供在线音乐欣赏功能，其中颇受用户欢迎的就是在线连续播放音乐，这样用户就可以一边工作一边欣赏自己喜欢的音乐了，既不耽误工作，又放松了心情。本实例将介绍如何实现在线连续播放音乐，运行本实例，在页面中将显示网站提供的歌曲列表，用户可以在自己喜欢的歌曲前面选中复选框，选择要播放的歌曲，也可以单击“全选”超链接或“反选”超链接，进行全选或反选，然后单击“连续播放”按钮，将打开歌曲连播页面，播放所选歌曲，如图 15.4 所示，还可以选择顺序播放或随机播放，默认情况下采用的是顺序播放。



图 15.4　顺序播放歌曲

关键技术

本实例中使用的技术要点大部分与媒体播放器有关，主要应用<object>标记调用 Windows Media Player 播放器，并通过 JavaScript 对播放器进行动态控制。通过 HTML 提供的<object>标记可以调用指定媒体播放器控件来

播放多媒体文件，<object>的语法格式如下：

```
<object classid="clsid" id="id" width="width" height="height"></object>
```

参数说明

❶ classid：用于指定使用的浏览器插件，如调用 Windows Media Play 播放器，可以将 classid 属性的值设置为 clsid:22D6F312-B0F6-11D0-94AB-0080C74C7E95。

❷ id：用于指定该对象的 id 值。

❸ width：用于指定媒体播放器的宽度。

❹ height：用于指定媒体播放器的高度。

在<object>标记中，还可以包括<param>子标记，该标记用于设置<object>标记所调用的媒体播放器的相关属性，比较常用的属性介绍如下。

- url：用于指定要播放文件的路径，可以是绝对路径也可以是相对路径。
- volume：用于控制音量，值为 0～100 之间的整数，表示 0%～100%。
- playcount：用于指定播放次数。
- enableerrordialogs：用于指定是否启用错误提示报告。
- autostart：用于指定是否自动播放，1 表示自动播放，0 表示不自动播放。

本实例通过调用 Windows Meida Player 播放器的具体代码如下：

```
<object classid='clsid:22D6F312-B0F6-11D0-94AB-0080C74C7E95' id='wghMediaPlayer' name='wghMediaPlayer' width='360' height='64'>
<param name='playcount' value='100'><param name='enablecontextmenu' value='0'>
<param name='enableerrordialogs' value='0'>
<param name='ShowStatusBar' value='-1'>
</object>
```

在 JavaScript 中，可以通过以下属性和方法对播放器进行动态控制，分别介绍如下。

- autoRewind 属性：用于设置当停止播放时是否返回到电影剪辑的开始部分；值为 true 时，表示返回到电影剪辑的开始部分，值为 false 时，表示不返回到电影剪辑的开始部分。
- sendPlayStateChangeEvents 属性：用于指定是否支持播放状态改变事件，值为 true 或 false。
- fileName 属性：用于指定要播放的文件。
- attachEvent()方法：用于为某一事件附加其他的处理事件。
- detachEvent()方法：用于拆分某一事件绑定的处理事件。
- play()方法：用于开始播放。
- stop()方法：用于停止播放。

设计过程

（1）创建 index.jsp 页面，通过 JSTL 标签将页面重定向到歌曲列表页面，关键代码如下：

```
<c:redirect url="/SongInfo">
    <c:param name="action" value="query"/>
</c:redirect>
```

（2）创建 songList.jsp 页面，在该页面中首先应用 JSTL 的<c:forEach>标签循环显示歌曲列表，然后在每首歌曲前面添加一个标记是否选中的复选框，关键代码如下：

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<c:forEach var="songForm" items="${list}">
  <tr>
      <td height="30" align="center" bgcolor="#FFFFFF"><input type="checkbox" class="noborder" id="playId" name="playId" value="
${songForm.id}"></td>
      <td bgcolor="#FFFFFF"> ${songForm.songName}</td>
      <td bgcolor="#FFFFFF"> ${songForm.singer}</td>
      <td bgcolor="#FFFFFF"> ${songForm.specialName}</td>
      <td bgcolor="#FFFFFF"> ${songForm.songType}</td>
  </tr>
</c:forEach>
```

（3）在 songList.jsp 页面的合适位置添加用于控制复选框全选或反选的复选框和“连续播放”超链接，关键代码如下：

```
<tr>
        <td width="88%" height="40" align="right" style="padding-right:10; color:#78A700">
        [<a style="cursor:hand;" onClick="CheckAll(form1.playId)">全选</a>/<a style="cursor:hand;" onClick="contraCheck(form1.playId)">反选]
    <div id="ch" style="display:none">
        <input name="playId" type="checkbox" class="noborder" value="0">
    </div></td>
<td width="12%" align="right" style="padding-right:10px"><a style="cursor:hand;" onClick="continuePlay(form1.playId,form1)"><img src="images/button_c.gif" width="60" height="17" border="0"></a></td>
<!--层用于放置隐藏的 checkbox 控件，因为当表单中只有一个 checkbox 控件时，应用 JavaScript 获得其 length 属性值为 undefine-->
    </tr>
```

（4）在 songList.jsp 页面的<head>标记中，编写自定义的 JavaScript 函数 CheckAll()，用于控制复选框的全选，具体代码如下：

```
//控制复选框的全选操作
function CheckAll(elementsA){
 for(i=0;i<elementsA.length;i++){
        elementsA[i].checked = true;                              //设置复选框为选中状态
 }
}
```

（5）在 songList.jgp 页面的<head>标记中，编写自定义的 JavaScript 函数 contraCheck()，用于控制复选框的反选，具体代码如下：

```
function contraCheck(elementsA){                                  //控制复选框的反选操作

 for(i=0;i<elementsA.length;i++){
            if(elementsA[i].checked){
                elementsA[i].checked = false;                     //设置复选框为非选中状态
            }else{
                elementsA[i].checked = true;                      //设置复选框为选中状态
            }
 }
}
```

（6）在 songList.jsp 页面的<head>标记中，编写自定义的 JavaScript 函数 continuePlay()，用于判断用户是否选择了要播放的歌曲，如果没有选择，则提示“请选择要播放的歌曲！”；否则提交表单进行播放。continuePlay()函数的具体代码如下：

```
function continuePlay(playId,formname){
 var flag = false;
 for(i=0;i<playId.length;i++){
        if(playId[i].checked){
                flag = true;
                break;                                            //跳出循环
        }
 }
 if(!flag){
        alert("请选择要播放的歌曲！");
        return false;
 }else{
        formname.submit();                                        //提交表单
 }
}
```

（7）在歌曲信息相关的 Servlet 中，编写实现歌曲连播的方法 continuePlay()。在该方法中，首先获取要进行连续播放歌曲的 ID，并将获取的 ID 数组连接为一个以逗号分隔的字符串，然后从数据库中查询要播放歌曲的信息，并保存到 HttpServletRequest 对象中，最后将页面重定向到歌曲连播页面。continuePlay()方法的具体代码如下：

```
    public void continuePlay(HttpServletRequest request,HttpServletResponse response) throws ServletException,IOException{
        ConnDB conn = new ConnDB();
        response.setContentType("text/html;charset=UTF-8");
        String ID[]=request.getParameterValues("playId");         //获取要播放的歌曲 ID
        String playID = "";
      for (int i = 0; i < ID.length; i++) {
            playID = playID + ID[i] + ",";
      }
        playID = playID.substring(0, playID.length() - 1);        //去除最后一个逗号
        String sql = "SELECT * FROM tb_song WHERE id IN ("+playID+")";
```

```
            System.out.println("SQL："+sql);
            ResultSet rs = conn.executeQuery(sql);                          //执行查询语句
            List list = new ArrayList();
            try {
                                                                            //获取歌曲文件的 URL 地址
                String url = request.getRequestURL().toString();
                url = url.substring(0, url.lastIndexOf("/") + 1) + "music/";
                while (rs.next()) {
                    SongForm f = new SongForm();
                    f.setSongName(rs.getString("songName"));                //歌曲名
                    f.setFileName(url+rs.getString("fileName"));            //歌曲文件的 URL 地址
                    list.add(f);
                }
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
            request.setAttribute("list", list);
            request.getRequestDispatcher("continuePlay.jsp").forward(request,
                    response);
        }
```

（8）编写歌曲连播页面 continueplay.jsp，实现按顺序或随机方式播放歌曲，在页面的合适位置添加一个表单及一张 3 行 2 列的表格，并在该表格的第一行的左侧单元格中输入提示性文字“随机播放”，在右侧的单元格中添加一个名称为 playType 的下拉列表框，该下拉列表框包括“顺序播放”和“随机播放”两个选项；将表格的第 3 行合并为一个单元格，并在该单元格中添加一个用于显示播放列表的列表框，关键代码如下：

```
<form name="form1" method="post" action="">
    <table width="363" height="185" border="0" cellpadding="0" cellspacing="0">
        <tr>
            <td colspan="2" id="myPlayer">正在加载播放器……</td>
              </tr>
        <tr>
             <td width="60" height="35">播放列表</td>
             <td width="303" align="right">
<select name="playType" id="playType">
                    <option value="0" selected>顺序播放</option>
                    <option value="1">随机播放</option>
                </select></td>
        </tr>
        <tr>
             <td colspan="2">
<select name="playList" size="10" id="playList" ondblclick="list_dblClick();" style="width:360px">
             <c:forEach var="songForm" items="${list}">
                    <option value="${songForm.fileName}">
                                ${songForm.songName}</option>
             </c:forEach>
     </select></td>
       </tr>
    </table>
</form>
```

（9）编写自定义的 JavaScript 函数 init()，用于在页面加载后调用 Media Player 播放器，按顺序方式播放歌曲列表，关键代码如下：

```
function init(){
 document.getElementById("myPlayer").innerHTML="<object  classid='clsid:22D6F312-B0F6-11D0-94AB-0080C74C7E95'  id='wghMediaPlayer'  name='wghMediaPlayer'  width='360'  height='64'><param  name='playcount'  value='100'><param  name='enablecontextmenu'  value='0'><param  name='enableerrordialogs' value='0'><param name='ShowStatusBar' value='-1'></object> ";
 document.getElementById("wghMediaPlayer").AutoRewind=false;           //设置当停止播放时不返回到电影剪辑的开始部分
 document.getElementById("wghMediaPlayer").AutoStart=true;             //设置自动播放
 document.getElementById("wghMediaPlayer").SendPlayStateChangeEvents=true;
//绑定播放状态改变事件
 document.getElementById("wghMediaPlayer").attachEvent("PlayStateChange",checkPlayStatus);
if(form1.playList.options.length>0){
        form1.playList.options[0].selected=true;                       //设置列表框的第一个选项为选中状态
        document.getElementById("wghMediaPlayer").fileName=form1.playList.value;   //指定歌曲文件
        document.getElementById("wghMediaPlayer").play();              //开始播放
 }
}
```

（10）编写自定义的 JavaScript 函数 checkPlayStatus()，用于当播放状态改变时连续播放歌曲，关键代码如下：

```
function checkPlayStatus(){
 try{
        if(document.getElementById("wghMediaPlayer").PlayState==0){                    //当播放状态为停止时
                document.getElementById("wghMediaPlayer").detachEvent("PlayStateChange",checkPlayStatus);
                                                                                       //拆分播放状态改变事件
                document.getElementById("wghMediaPlayer").stop();                      //停止播放
        if(form1.playType.value==0){                                                   //表示顺序播放
                if(form1.playList.options.selectedIndex<form1.playList.options.length-1){
                        form1.playList.options[form1.playList.options.selectedIndex+1].selected=true;
                }else{
                        form1.playList.options[0].selected=true;                       //设置列表框的第一个选项为选中状态
                }
        }else{                                                                         //随机播放
                var randomValue=Math.round(Math.random() * (form1.playList.options.length - 1)) ;  //生成一个 0 至歌曲总数-1 的随机整数
                form1.playList.options[randomValue].selected=true;
        }
                document.getElementById("wghMediaPlayer").fileName=form1.playList.value;
                document.getElementById("wghMediaPlayer").play();                      //开始播放
 setTimeout('document.getElementById("wghMediaPlayer").play();document.getElementById("wghMediaPlayer").attachEvent("PlayStateChange",chec
kPlayStatus);',1000);
        }
 }catch(e){
        alert("出错了");
```

（11）在进行歌曲连播时，为了让用户可以选择指定的歌曲开始播放，还需要双击列表框，在指定歌曲时添加开始播放的功能，实现该功能时，需要编写一个自定义的 JavaScript 函数，这里为 list_dblClick()，在该函数中，也需要根据选择的播放器从指定的歌曲开始播放。list_dblClick()函数的具体代码如下：

```
function list_dblClick(){
//拆分播放状态改变事件
 document.getElementById("wghMediaPlayer").detachEvent("PlayStateChange",checkPlayStatus);
 document.getElementById("wghMediaPlayer").fileName=form1.playList.value;     //将列表框的值指定给 Media Player 播放器
 document.getElementById("wghMediaPlayer").play();                            //开始播放
 setTimeout('document.getElementById("wghMediaPlayer").play();document.getElementById("wghMediaPlayer").attachEvent("PlayStateChange",chec
kPlayStatus);',1000);
}
```

list_dblClick()函数编写完成后，还需要在列表框的 ondbclick 事件中调用方法。

秘笈心法

在连续播放音乐时，通过 JavaScript 脚本实现歌曲连续播放，首先使用 document 对象的 getElementById()方法获取播放音乐的状态，如是否为播放状态、停止播放等。再在 continueplay.jsp 中设置歌曲播放的初始化方法 init()和当播放状态改变时执行的方法 checkPlayStatus()，而在实现以上方法时首先要使用 innerHTML 获取 HTML 中的内容。

实例 410	同步显示 LRC 歌词 光盘位置：光盘\MR\15\410	高级 实用指数：★★★★

实例说明

在音乐网站开发过程中，如果在试听歌曲模块中加入同步显示 LRC 歌词功能，能够为网站增色不少。本实例将介绍同步显示 LRC 歌词的方法。运行本实例，在页面中将显示网站提供的歌曲列表，单击某一首歌曲后面的“播放”超链接，将打开在线试听页面并播放该歌曲。如果该歌曲提供了 LRC 歌词文件，还将同步显示歌词，运行结果如图 15.5 所示。

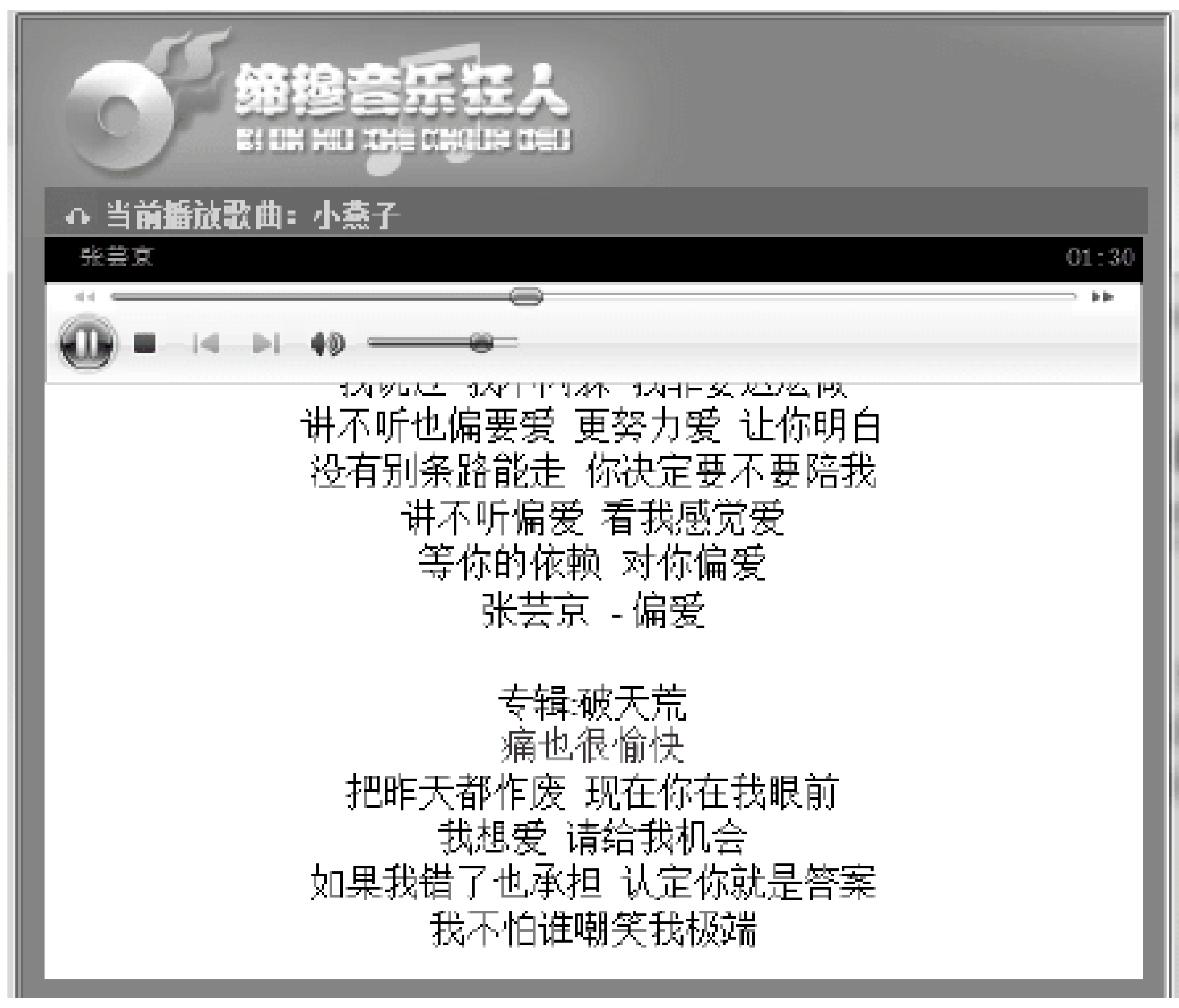


图 15.5　同步显示歌词

关键技术

本实例主要涉及 LRC 歌词的格式和读取 LRC 歌词的行数两个技术要点。首先是 LRC 歌词的格式，在 LRC 歌词中，通过[MM:SS:MS]指定时间，通过[ar：演唱者名]指定演唱者，通过[ti：歌曲名]指定歌曲名，通过[al：专辑名]指定专辑名，通过[by：歌词编辑者]指定歌词编辑者，通过[Offset:MS]调整整个歌词文件的时间标签值。

其次是获取歌词的行数，由于在 LRC 歌词格式中每一个中括号对代表一行歌词，所以要获取歌词的行数，就需要将歌词以中括号为分隔进行分解。在 Java 中要将指定的字符串按某个分隔符进行分解，可以使用 java.util.StringTokenizer 类，该类用于分析字符串，并将字符串分解成可被独立使用的单词，StringTokenizer 类有两个常用的构造方法。

```
StringTokenizer(String s)
```

该构造方法为字符串 s 构造一个分析器，使用默认的分隔集合，即空格符（若干个空格被看作一个空格）、换行符、回车符、Tab 符。

```
StringTokenizer(String s,String delim)
```

该构造方法为字符串 s 构造一个分析器，参数 delim 被作为分隔符（此处支持正则表达式）。使用 StringTokenizer 类创建一个字符串分析器后，就可以使用 countTokens()方法获取字符串共有多少个语言符号，同时也可以使用 nextToken()方法逐个获取字符串中的语言符号（单词），每当调用 nextToken()时，都将在字符串中获得下一个语言符号。

设计过程

（1）创建 index.jsp 和 songList.jsp 页面，分别用于加载重定向歌曲列表到指定的页面中和使用<c:forEach>标签循环显示歌曲列表并在每首歌曲的后面添加“播放”超链接。

（2）在歌曲信息相关的 Servlet 中，编写实现在线试听的方法 play()，在该方法中首先获取要播放歌曲的 ID，并根据 ID 从数据库中获取该歌曲的信息，然后读取该歌曲对应的歌词文件（即 LRC 文件），将读取的歌词内容连接成一个字符串，并统计歌词的行数，再将歌词的行数、歌词内容、当前页的歌曲信息和当前试听的歌曲名称保存到 HttpServletRequest 对象中，最后将页面重定向到在线试听页面。play()方法的关键代码如下：

```
public void play(HttpServletRequest request, HttpServletResponse response) throws
            ServletException, IOException {
        ConnDB conn = new ConnDB();
        response.setContentType("text/html;charset=UTF-8");
        int ID = Integer.parseInt(request.getParameter("playId"));            //获取要播放的歌曲 ID
        String sql = "SELECT * FROM tb_song WHERE id ="+ID+"";
        System.out.println("SQL："+sql);
        ResultSet rs = conn.executeQuery(sql);                                 //执行查询语句
        String fileName="";                                                    //歌曲文件名
        String songName="";                                                    //歌曲名
        String url ="";                                                        //歌曲文件的 URL 地址
        try {
```

```
                                                                    //获取歌曲文件的 URL 地址
            url = request.getRequestURL().toString();
            url = url.substring(0, url.lastIndexOf("/") + 1) + "music/";
            if (rs.next()) {
                songName=rs.getString("songName");                  //歌曲名
                fileName=rs.getString("fileName");                  //歌曲文件名
            }
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
        /** **************获取歌词*********************** */
        String lrcRealPath = request.getRealPath("/");
        String mp3RealPath = url + fileName;
        lrcRealPath = lrcRealPath + "music/" +
                fileName.substring(0, fileName.lastIndexOf(".") + 1) + "lrc";   //lrc 文件路径
        String content = "";                                        //歌词内容
        int lineNumber = 0;                                         //歌词的行数
        File lrcFile = new File(lrcRealPath);                       //声明歌词文件对应的 File 对象
        if (lrcFile.exists()) {                                     //判断歌词文件是否存在
            FileInputStream lrcf;
            try {
                lrcf = new FileInputStream(lrcRealPath);
                int sl = 0;
                                                                    //available()方法可以不受阻塞地从此输入流中读取（或跳过）估
计剩余字节数
                byte[] data = new byte[lrcf.available()];
                while ((sl = lrcf.read(data)) > 0) {
                    content += new String(data, 0, sl);             //将歌词内容连接为一个字符串
                }
                                                                    //分析字符串中共包括多少个中括号对“[]”
                StringTokenizer st = new StringTokenizer(content, "\\[*\\]");
                lineNumber = st.countTokens();                      //返回分析的结果
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
            request.setAttribute("realPath", mp3RealPath);          //保存要播放歌曲的完整路径
        request.setAttribute("lineNumber", lineNumber);             //保存歌词的行数
        request.setAttribute("lrcContent", content);                //保存歌词的内容
        request.setAttribute("fileURL", fileName);                  //保存当前页的歌曲信息
        request.setAttribute("songName", songName);                 //保存当前试听的歌曲名称
        request.getRequestDispatcher("play.jsp").forward(request,response);
    }
```

（3）编写在线试听页面 play.jsp，实现播放歌曲并设置歌词同步显示。通过 EL 表达式中的${}将获取的歌词内容输出到 id 属性为 lrcContent 的<span>标记中，并设置该标签为不显示，关键代码如下：

```
<span id="lrcContent" style="display:none;">${lrcContent}</span>
```

在页面的合适位置添加一个用于显示歌词的<div>标记，并将该标记的 style 属性的子属性 overflow 设置为 hidden，即超出指定范围的内容隐藏，关键代码如下：

```
<div id="lrcAreaDiv" style="overflow:hidden; height:260; width:480; background-color:#FFFFFF">
```

在 id 为 lrcAreaDiv 的<div>标记中，添加一个 id 属性为 lrcArea 的表格，并根据获取的歌词行数生成指定行的单元格，关键代码如下：

```
<table border="0" cellspacing="0" cellpadding="0" id="lrcArea" width="100%" style="position:relative; top:120px;">
  <tr><td nowrap height="20" align="center">
    <table border="0" cellspacing="0" cellpadding="0">
      <tr><td nowrap height="20"><span id="lrcLine1" style="height:20; color:#FF0000">正在加载歌词……</span></td>
      </tr>
      <tr style="position:relative; top: -20px; z-index:6;">
        <td nowrap height="20"><div id="lrcLine_will1" class="lrcLine_will"></div></td>
      </tr>
    </table>
    </td></tr>
<c:forEach begin="0" end="${lineNumber}" step="1" var="i">
  <tr style="position:relative; top: ${-20*i}px;"><td nowrap height="20" align="center">
    <table border="0" cellspacing="0" cellpadding="0">
```

```
        <tr><td nowrap height="20"><span id="lrcLine${i+2}" style="height:20"></span></td></tr>
        <tr style="position:relative; top: -20px; z-index:6;">
          <td nowrap height="20"><div id="lrcLine_will${i+2}" class="lrcLine_will"></div></td>
        </tr>
      </table>
    </td></tr>
  </c:forEach>
</table>
```

（4）通过 JavaScript 解析歌词并控制歌词同步显示，当不存在歌词时显示提示信息“很抱歉，该歌曲没有提供歌词！”，关键代码如下：

```
<script language="JavaScript">
var getLrcContent=lrcContent.innerHTML;          //获取歌词内容
if(getLrcContent!=""){
lrcobj = new lrcClass(getLrcContent);            //初始化 lrcClass 类的对象，参数为歌词内容
var lrc0, lrc1;
moveflag = false;
movable = false;
moven = false;
var lrctop;
predlt = 0;
curdlt = 0;
curpot = 0;
//定义一个解析 lrc 歌词的函数
function lrcClass(lyric){                          //参数为歌词内容
        this.inr = [];
        this.oTime = 0;
        this.dts = -1;
        this.dte = -1;
        this.dlt = -1;
        this.ddh;
        this.fjh;
        //获取歌词中是否有时间补偿值，时间补偿值的单位为毫秒，正数表示整体提前，负数表示整体滞后
        if(/\[offset\:(\-?\d+)\]/i.test(lyric))
                this.oTime = RegExp.$1/1000;
        lyric = lyric.replace(/\[\:\][^$\n]*(\n|$)/g,"$1");
        lyric = lyric.replace(/\[[^\[\]\:]*\]/g,"");
        lyric = lyric.replace(/\[[^\[\]]*[^\[\]\d]+[^\[\]]*\:[^\[\]]*\]/g,"");
        lyric = lyric.replace(/\[[^\[\]]*\:[^\[\]]*[^\[\]\d\.]+[^\[\]]*\]/g,"");
        while(/\[[^\[\]]+\:[^\[\]]+\]/.test(lyric)){
                lyric = lyric.replace(/((\[[^\[\]]+\:[^\[\]]+\])+[^\[\r\n]*)[^\[]*/,"\n");
                var zzzt = RegExp.$1;
                /^(.+\])([^\]]*)$/.exec(zzzt);
                var ltxt = RegExp.$2;
                var eft = RegExp.$1.slice(1,-1).split("][");
                for(var ii=0; ii<eft.length; ii++){
                        var sf = eft[ii].split(":");
                        var tse = parseInt(sf[0],10) * 60 + parseFloat(sf[1]);
                        var sso = { t:[] , w:[] , n:ltxt }
                        sso.t[0] = tse-this.oTime;
                        this.inr[this.inr.length] = sso;
                }
        }
//开始播放歌词的方法
function wghLoad_lrc(){
        lrcobj.wghLoad(mediaPlayer.controls.currentPosition);
        if(arguments.length==0){
                lrc0 = window.setTimeout("wghLoad_lrc()",10);
        }
}
//当页面卸载时，取消对 lrc0 的延迟执行
window.onunload=function(){
        clearTimeout(lrc0);
}
//设置歌词的顶部位置
function lrcTopPosition(nline){
        lrctop -= 20*nline;
        lrcArea.style.top = lrctop;
}
```

```
//改变歌词顶部的位置，实现歌词向上滚动
function lrcChangePosition(step,dur){
	if(moveflag) return;
	lrcArea.style.top = lrctop--;
	if(step<20){
		step++;
		window.setTimeout("lrcChangePosition("+step+","+dur+");",dur*50);
	}
}
//设置当前演唱的歌词行的颜色，即让当前歌词行高亮显示
function highlight(lid){
	lid.style.color = "#FF0000";			//设置将要演唱的歌词的颜色
}
//清除当前歌词的高亮显示
function loseColor(lid){
	window.clearTimeout(lrc1);
}
//演唱后的歌词行的颜色
function loseLight(lid){
	lid.style.color = "#000000";			//设置演唱后的歌词的显示颜色
}
wghLoad_lrc();						//开始播放歌词
}else{
document.getElementById("lrcLine1").innerHTML="很抱歉，该歌曲没有提供歌词！";
}
</script>
```

秘笈心法

在 play.jsp 页面中，实现同步显示 LRC 歌词时使用了 JavaScript 脚本，通过 innerHTML 获取歌词的内容，并对其 lrcCLass 类进行初始化，它的参数为歌词内容，并利用一些算法控制歌词的同步显示。也就是说获取歌词中是否有时间补偿值，时间补偿值的单位为毫秒，正数表示整体提前，而负数表示整体滞后，也就是所谓歌词延迟设置。

实例 411	把显示后的 LRC 歌词变换颜色 光盘位置：光盘\MR\15\411	高级 实用指数：★★★★

实例说明

在网站播放歌词时，经常是播放到哪句歌词就更改哪句歌词的颜色，以达到提示的作用。本实例是为了更加突出显示同步播放歌词，更改播放后歌词的颜色，运行结果如图 15.6 所示。



图 15.6　播放后的歌词颜色

关键技术

本实例在实现播放歌词时，应用了 JavaScript 脚本自定义的 highLight()和 loseLight()函数，用于当前演唱歌词行的颜色，即让当前歌词行高亮显示和演唱后的歌词的显示颜色，其中使用了 style 样式的 color 颜色属性，设置相应歌词状态的颜色，关键语句如下：

```
lid.style.color = "lightblue";
```

设计过程

（1）编写数据库的类及歌曲相关信息的 Servlet 实现类 SynchroLRC，这部分不是本实例介绍的重点，这里不再介绍，具体代码请参见光盘。

（2）编写 index.jsp 页面，在该页面中使用 JSTL 标签把页面重定向到歌曲列表页面，代码如下：

```
<c:redirect url="/SynchroLRC">
    <c:param name="action" value="query"/>
</c:redirect>
```

（3）编写在线试听 play.jsp 页面，通过 JavaScript 脚本实现播放歌曲并设置播放后歌词颜色为 lightblue，关键代码如下：

```
//设置当前演唱的歌词行的颜色，即让当前歌词行高亮显示
function highlight(lid){
        lid.style.color = "#FF0000";            //设置将要演唱的歌词的颜色
}
//清除当前歌词的高亮显示
function loseColor(lid){
        window.clearTimeout(lrc1);
}
//演唱后的歌词行的颜色
function loseLight(lid){
        lid.style.color = "lightblue";          //设置演唱后的歌词的显示颜色
}
wghLoad_lrc();                                  //开始播放歌词
}else{
document.getElementById("lrcLine1").innerHTML="很抱歉，该歌曲没有提供歌词！ ";
}
</script>
```

秘笈心法

在线播放歌曲并同步显示歌词时，通过直接在页面中使用 JavaScript 脚本可以方便快速地实现，因为 JavaScript 的灵活性以及可扩展性都很高。通过本实例的实现，读者可以尝试实现在播放每个字时改变歌词颜色。

15.2　插入 Flash 动画

Flash 是一种矢量格式的动画文件，可以包含动画、声音、超文本链接等，而且文件的体积也很小，如果将其嵌入到网页中，那么整个网页会增色不少。本节将通过两个典型实例介绍 Flash 动画在网页中的应用。

实例 412	插入 Flash 动画 光盘位置：光盘\MR\15\412	高级 实用指数：★★★★

实例说明

为了美化网站，使其有一个更好的视觉感受，可以将 Flash 技术引入到网页中，使网页更具有表现力。打开大多数网页，都会不断弹出各种各样用 Flash 制作的精美动感广告和有趣的 Flash 游戏或 MTV，增强了网页的宣传

效果。由此可见，Flash 技术在网页中起着举足轻重的作用。将 Flash 动画插入到网页中的效果如图 15.7 所示。



图 15.7　插入 Flash 动画

关键技术

本实例主要应用了<object>标记和<embed>标记。<object>标记中的 classid 是告诉浏览器插件的类型，codebas 是可选参数；安装 Flash 插件的用户浏览网页时，自动连接到 Shockwave 的下载网页，自动下载并安装相关插件。<object>和<embed>标记中 quality = high 的记号作用是使浏览器以高质量浏览动画。

设计过程

创建 index.jsp 页面，在该页面中定义<object>标签和<embed>标签，关键代码如下：

```
<objectclassid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=7,0,19,0" width="830" height="471">
   <param name="movie" value="flash/flash.swf" />
   <param name="quality" value="high" />
   <param name="wmode" value="transparent" />
   <embed src="flash.swf" width="830" height="471" quality="high" pluginspage="http://www.macromedia.com/go/getflashplayer" type="application/x-shockwave-flash" wmode="transparent"></embed>
</object>
```

秘笈心法

<object>标签的 classid 属性是指定浏览器所有的 ActiveX 控件，要注意的是仅用于 object 中。而<embed>是用来播放影音文档的，通常播放 Windows Media Player 格式，也可以播放其他格式，它支持插入的多媒体可以是 MIDI、WAV、AIFF、AU、MP3、MPEG、AVI 等。

实例 413	插入背景透明的 Flash 动画 光盘位置：光盘\MR\15\413	高级 实用指数：★★★★

实例说明

如果说在网页中插入 Flash 是为了美化网站，使网页更具有表现力，那么在网页中插入背景透明的 Flash，则会使网站的整体效果更加和谐统一，达到更完美的效果。运行本实例，该网站顶部的企业文化的背景动画就是插入的透明背景的 Flash，如图 15.8 所示。



图 15.8　插入背景透明的 Flash 动画

关键技术

在网页中插入背景透明的 Flash 动画，主要是通过设置<object></object>或<embed></embed>标记的 wmode 属性来实现的。

在<object></object>标记之间加入如下代码：

```
<param name="wmode" value="transparent">
```

设置<embed></embed>标记的 wmode 属性的代码如下：

```
wmode="transparent"
```

设计过程

创建 index.jsp 页面文件，并在页面中将 Flash 动画文件添加到页面中的代码如下：

```
<objectclassid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=7,0,19,0" width="551" height="143">
        <param name="movie" value="tm.swf" />
        <param name="quality" value="high" />
      <param name="wmode" value="transparent">
        <embed src="tm.swf" quality="high" pluginspage="http://www.macromedia.com/go/getflashplayer" type="application/x-shockwave-flash"
width="551" height="143" wmode="transparent"></embed>
</object>
```

秘笈心法

在<object>中使用了 codebase 属性，它是一个可选属性，用于提供一个基本的 URL，该属性的值是一个 URL，指向的目录包含了 classid 属性来引用对象，codebase URL 会覆盖文档的基本 URL，但不会永久替代它，如果不使用 codebase 属性，这个基本的 URL 就是默认的。

15.3 播 放 视 频

实例 414	播放视频文件 光盘位置：光盘\MR\15\414	高级 实用指数：★★★★

实例说明

在网站中添加视频文件是很常见的，有了在线播放视频文件，不仅可增加相关信息的生动性，还可以为网站增加人气。目前，有很多在线视频网站，比较典型的如优酷网和土豆网。运行本实例，加载完页面后将自动播放视频，程序运行结果如图 15.9 所示。



图 15.9 播放 AVI 文件

关键技术

在 HTML 文件中可以直接嵌入多媒体文件。多媒体是指利用计算机技术，把多种媒体综合在一起，使之建

立起逻辑上的联系，并能对其进行各种处理的一种方法。多媒体主要包括文字、声音、图像和动画等各种形式。在 HTML 文件中使用标记<embed>嵌入多媒体文件，其语法格式如下：

```
<embed SRC="file_url" WIDTH=value HEIGHT=value HIDDEN=hidden_value AUTOSTART=auto_value LOOP=loop_value>
</embed>
```

标记<embed>的属性如表 15.4 所示。

表 15.4　标记<embed>的属性

属　性	描　述
SRC	多媒体文件路径
WIDTH	播放多媒体文件区域的宽度
HEIGHT	播放多媒体文件区域的高度
HIDDEN	控制播放面板的显示和隐藏，取值为 true 代表隐藏面板，取值为 no 代表显示面板
AUTOSTART	控制多媒体内容是否自动播放，取值为 true 代表自动播放，取值为 false 代表不自动播放
LOOP	控制多媒体内容是否循环播放，取值为 true 代表无限次循环播放，取值为 no 代表仅播放一次

设计过程

（1）创建 index.jsp 页面，根据需要在指定的页面中添加表格或表单元素，关键代码如下：

```
<table width="803" border="0" align="center" cellpadding="0" cellspacing="0">
  <tr>
    <td width="803" height="75" valign="top" background="images/top.jpg"><table width="100%"   border="0" cellspacing="0" cellpadding="0">
      <tr>
        <td height="30"><table width="100%" cellspacing="0" cellpadding="0">
          <tr align="center" valign="bottom">
            <td width="78%" height="29" align="right" class="style2">设为首页</td>
            <td width="9%" align="right" class="style2"><span class="style2"> | </span>收藏本站</td>
            <td width="11%"><span class="style2"> | 联系我们</span></td>
            <td width="2%"> </td>
          </tr>
        </table></td>
      </tr>
      <tr>
        <td> </td>
      </tr>
    </table></td>
  </tr>
  <tr>
    <td><table width="100%"   border="0" cellspacing="0" cellpadding="0">
      <tr>
        <td width="803" height="166" background="images/banner.jpg"> </td>
      </tr>
    </table></td>
  </tr>
  <tr>
    <td height="30"><table width="100%"   border="0" cellspacing="0" cellpadding="0">
      <tr>
        <td width="803" height="38" background="images/dh.jpg"><div align="center"></div></td>
      </tr>
    </table></td>
  </tr>
</table>
```

（2）在网页中指定的位置处插入<embed>标记，通过该标记实现 AVI 文件的播放，将<embed>标记中的 Loop 参数值设为 true，表示指定的 AVI 文件进行循环播放，关键代码如下：

```
<embed src="images/new.swf" width="30" height="18" align="absbottom"
 noerror="true" loop="true" autostart="true"   ></embed>
```

秘笈心法

如果使用<object>标记的同时也使用<embed>标记，则对每个属性或参数都要使用相同的值，以确保能在各

种浏览器中进行一致的回放。而 type 值对应相应的播放器，Windows Media Player 7 以上版本插件为 application/x-mplayer2，Realplayer 播放器及其浏览插件为 audio/x-pn-realaudio-plugin，QuickTime 播放器为 video/quickTime。

实例 415 自制视频播放器

高级

光盘位置：光盘\MR\15\415

实用指数：★★★★

实例说明

AVI 是比较常用的多媒体视频文件格式，利用多媒体制作软件可以制作出许多丰富多彩的视频文件，但是制作好的 AVI 视频文件如何在网络中播放呢？通常有两种方法，一是应用超链接标签来播放，应用该方法时，系统会利用默认的播放器播放该文件；另一种方法是应用 HTML 的标记符<embed>来播放。本实例使用的就是后一种方法，结合图层制作了一个多媒体播放器。运行程序，单击“单击观看多媒体演示”图标，将打开如图 15.10 所示的窗口播放 AVI 文件。



图 15.10　自制视频播放器

关键技术

实现本实例主要利用 HTML 的标记符<embed>直接调用 Media Player 播放 AVI 视频文件，但是其界面并不是很美观，尤其是底部的播放控制区不可以隐藏，这时可以通过在页面中加入一个浮动的图层，并使用该图层遮盖住系统播放器的灰色控制区来实现隐藏。

在 IE 浏览器中，层依靠<DIV></DIV>和<SPAN></SPAN>来实现，两者基本相同。通常<DIV>用于较大的层，<SPAN>用于较小的层，并且<DIV>在实现的层后面加上一个回车键换行，而<SPAN>不加。本实例在网页中加入浮动的图层利用<DIV>实现。

设计过程

（1）在网页中适当的位置添加“单击观看多媒体演示”的图片，并为该图片添加超链接，单击超链接所执行的操作时打开新窗口，用于播放 AVI 视频文件，关键代码如下：

```
<div align="center" class="style3">单击观看多媒体演示：</div>
<ahref="#"onClick="JavaScript:window.open('AVIPlay.htm',",'width=400,height=352')"><img src="image/flag.jpg" width="105" height="54" border="0"></a>
```

（2）利用 HTML 的标记符<embed></embed>播放 AVI 视频文件的关键代码如下：

```
<embed src="AVI/aa.avi" width="326" height="315" noerror="true" type="video/avi">
```

（3）为了隐藏其底部的播放控制区可以应用图层，利用图层将其底部的播放控制区遮盖住，以达到“天衣无缝”的效果，关键代码如下：.

```
<div id="Layer1" style="position:absolute; width:199px; height:60px; z-index:1; left: 0px; top: 292px;">
<img src="image/play_02.gif" width="410" height="60"></div>
```

秘笈心法

DIV 的语法如下：

```
<div id=layername style="style definition">
layer content
</div>
```

其中 style definition 是一组用分号隔开的样式定义。常用的有以下几种。

❶ position：其值可以是 absolute 和 relative，所谓的 absolute（绝对定位）就是层的内容不必按照出现的先后次序在浏览器上依次排列，而是可以以像素为单位定位到浏览器用户区域的任意位置；而 relative（相对定位）则是层按照相邻的内容依次排行。

❷ left,top,width,height：指层的左上角坐标以及它的宽度和高度。

❸ z-index：由于层可以被放置在页面的任何位置，当它与其他内容重合时，z-index 值大的显示在上面，所有层的 z-index 值为 1。但是在 NS 中，所有的表单元素（如文本框、列表框、按钮等），只要是可见的，就无法被遮住，无论 z-index 的值是多少。因此在设计页面时，要注意不要在动态出现或隐藏的元素（如弹出式菜单）的位置上放置表单元素。

❹ visibility：指明层是否可见，通过在程序中动态改变这个值，可以实现层的出现和消失，如下拉菜单就要依靠它来实现。它的 3 个候选值为 inherit，可见性与父元素的可见性相同；visible，可见（在 NS 中是 show）；hidden，不可见（在 NS 中是 hide）。

实例 416	在线播放 FLV 视频 光盘位置：光盘\MR\15\416	高级 实用指数：★★★★

实例说明

FLV 流媒体格式是一种新的视频格式，全称为 Flash Video，由于其文件体积小巧、加载速度快和视频质量好等特点，使其在网站上迅速盛行，目前各在线视频网站均采用此视频格式，如新浪播客等，FLV 已经成为当前视频文件的主流格式。运行本实例，在页面中将显示网站提供的视频列表，在该页面中单击某一视频后面的“观看”超链接，将打开播放 FLV 视频页面，在该页面中将调用 FLV 播放器，单击“单击开始播放”超链接，将开始播放选择的视频，如图 15.11 所示，在播放的过程中，用户可以通过“播放”按钮、“暂停”按钮和“停止”按钮对视频的播放进行控制。



图 15.11　在线播放 FLV 视频

关键技术

本实例建立了一个 FlvInfo.java 的 Servlet，在其中实现了查询 FLV 视频详细信息 query()方法和播放 FLV 视频 play()方法，再通过 JSP 页面，调用 Flash 中制作的 FLV 播放器实现在线播放 FLV 视频。FLV 播放器实际上是一个扩展名为 swf 的 Flash 文件，在 JSP 中，可以通过 HTML 提供的<object>标记和<embed>标记进行调用，通过<object>标记调用 Flash 播放器的基本代码如下：

```
<objectclassid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=7,0,19,0"width="450" height="300">
        <param name="movie" value="${url}player.swf?fileName=${url}${fileName}"/>
        <param name="quality" value="high">
</object>
```

参数说明

❶ codebase 属性：用于指定获取组件的 URL。

❷ movie 参数：用于指定要播放文件的 URL。

❸ quality 参数：用于指定播放资源的质量。

设计过程

（1）创建 index.jsp 页面，在该页面中通过 JSTL 标签将页面重定向到视频列表页面，代码中“/FlvInfo”指定的是 Servlet 映射路径，是通过 Servlet 将页面重定向到视频列表页面 flvList.jsp，关键代码如下：

```
<c:redirect url="/FlvInfo">
     <c:param name="action" value="query"/>
</c:redirect>
```

（2）创建 flvList.jsp 页面，在该页面中，首先应用 JSTL 的<c:forEach>标签循环显示视频列表，然后在每个视频后面添加“观看”超链接，关键代码如下：

```
<c:forEach var="flvForm" items="${list}">
          <tr>
               <td height="27" align="left" bgcolor="#FFFFFF"> ${flvForm.title}</td>
               <td align="left" bgcolor="#FFFFFF"> ${flvForm.introduce}</td>
               <td align="center" bgcolor="#FFFFFF"><a href="FlvInfo?action=play&playId=${flvForm.id}" target="_blank">观看</a></td>
          </tr>
</c:forEach>
```

（3）在 Flash 中制作一简单的 FLV 播放器，然后编写 FLV 播放器播放指定 FLV 视频文件的页面 play.jsp，关键代码如下：

```
<objectclassid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=7,0,19,0" width="450" height="300">
               <param name="movie" value="${url}player.swf?fileName=${url}${fileName}"/>
               <param name="quality" value="high">
               <embedsrc="${url}player.swf?fileName=${url}${fileName}"quality="high"
pluginspage="http://www.macromedia.com/go/getflashplayer"type="application/x-shockwave-flash" width="450" height="300" ></embed>
               </object>
```

（4）在 play.jsp 页面中调用 FLV 播放器，具体代码如下：

```
<objectclassid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=7,0,19,0" width="450" height="300">
                    <param name="movie" value="${url}player.swf?fileName=${url}${fileName}"/>
                    <param name="quality" value="high">
                    <embed src="${url}player.swf?fileName=${url}${fileName}" quality="high" pluginspage="http://www.macromedia.com/go/getflashplayer"type=
"application/x-shockwave-flash" width="450" height="300" ></embed>
               </object>
```

秘笈心法

本实例在 play.jsp 页面中调用 Flash 8 制作的 FLV 播放器时，使用<object>和<embed>标签，其中<embed>有一个 pluginspage 属性，它是标识 Flash Player 插件的位置，在尚未安装该插件时用户可以下载到它，仅用于<embed>标签中。

第 4 篇

窗体应用篇

⏭ 第 16 章　窗口的应用

⏭ 第 17 章　导航条的应用

⏭ 第 18 章　表单的应用

⏭ 第 19 章　表格的操作

第16章

窗口的应用

- 弹出窗口控制
- 弹出网页对话框
- 窗口的动画效果
- 窗口控制
- 框架的应用
- 无边框窗口

16.1　弹出窗口控制

本节所讲述的弹出窗口是指通过 window.open()方法打开新的窗口，这种弹出的窗口在做网页时是经常被用到的，如弹出一些网页的公告和一些商业广告，以及一些警告信息等。下面将通过一些实例介绍如何控制弹出窗口，使制作的网页有更好的体验。

实例 417	打开网页显示广告信息 光盘位置：光盘\MR\16\417	初级 实用指数：★★★★

■ 实例说明

广告是网站最大的盈利手段，任何网站都会推出一些用于广告的版块。但有时过多地在网页中分割出广告版块，会降低用户对网站的使用率。那么该如何解决这个问题呢？这时可以通过打开新的窗口来显示网站的广告信息。本实例就将实现在用户打开窗口时自动地弹出新窗口，而且用户可以随时关掉弹出窗口效果。实例运行效果如图 16.1 所示。



图 16.1　打开网页显示广告信息

■ 关键技术

本实例主要应用 JavaScript 脚本语言中 window 对象的 open()方法来实现。在 JavaScript 语言中 window 对象代表的是一个 Web 浏览器窗口或者是窗口中的一个框架，可以使用 window 对象对 Web 浏览器窗口或窗口中的框架进行控制。window 对象的常用方法如下。

- ❑ alert()方法：弹出警告对话框。
- ❑ close()方法：关闭被引用的窗口。
- ❑ confirm()方法：弹出确认对话框。
- ❑ focus()方法：将被引用的窗口放在所有打开窗口的前面。
- ❑ open()方法：打开浏览器窗口并显示由 URL 或名称引用的文档，再设置创建窗口的属性。
- ❑ prompt()方法：弹出一个提示对话框。
- ❑ print()方法：打印窗口或框架中的内容。
- ❑ resizeTo(x,y)方法：将窗口的大小设置为(x,y)，x、y 分别是宽度和高度。
- ❑ resizeBy(offsetx,offsety)方法：按照指定的位移量设置窗口大小，当 offsetx、offsety 的值大于 0 时为扩大，小于 0 时为缩小。

下面对本实例中应用到的 open()方法进行详细说明。

window 对象的 open()方法用于打开浏览器的窗口。其使用语法如下：

```
windowVar=window.open(url,windowname[,location]);
```

设计过程

（1）首先创建用于弹出广告信息的页面 new.htm，在该页面中添加所要宣传的广告信息。

（2）在网站首页 index.htm 设置每次进入网站首页时将弹出广告窗口，代码如下：

```
<script language="javascript">
        window.open("new.htm","new","height=141,width=600,top=10,left=20");             //打开新窗口
</script>
```

秘笈心法

弹出窗口可以应用到很多性质的网站，而且这些弹出窗口有着很大的宣传性，做好弹出窗口的控制和设计，可以为网站创造巨大的利益。

实例 418 定时关闭广告窗口

光盘位置：光盘\MR\16\418

初级

实用指数：★★★☆

实例说明

很多网站在浏览者进入网站以后会弹出新的窗口，这些窗口的内容一般是广告或网站服务协议等。大多数情况下，这些新打开的窗口需要浏览者自己去关闭，这样就给浏览者带来很大的不便，最终会影响到网站的访问量，而本实例将实现自动关闭新弹出的窗口。运行本实例一定时间后，新弹出的窗口就会自动关闭，效果如图 16.2 所示。



图 16.2　自动关闭广告窗口

关键技术

实现本实例主要是应用了 window 对象的 setTimeout()方法和 close()方法。使用 window 对象中的 setTimeout()方法主要是起到延时执行某一操作的作用，其语法格式如下：

```
setTimeout(expression,secdlay[,language])
```

参数说明

❶ expression：是一个字符串参数，用来调用任何函数、方法以及 JavaScript 语句。

❷ secdlay：指定运行的时间，以毫秒为单位。

❸ language：指定语句或参数 expression 调用的函数所使用的语言。如果都为 JavaScript 脚本语言，那么可以不用设置。

设计过程

（1）创建网站首页 index.htm，编写弹出窗口控制代码，关键代码如下：

```
<script language="javascript">
<!--
 window.open("guanbi.htm","advertise","width=557,height=176,top=10,left=20");
-->
</script>
```

（2）创建弹出窗口 guanbi.htm，并编写弹出窗口中的自动关闭控制代码，关键代码如下：

```
<body onload="window.setTimeout('window.close()',5000)">
```

秘笈心法

本实例的实现内容可以应用到很多行业类型的网站，如教育网站的一些紧急通知、商业网站的广告，也可应用于信息网站的信息服务协议。读者可以试着拓宽其使用范围，也可以试着编写如何定时打开一个窗口，方法是一样的。

实例 419	弹出窗口的居中显示 光盘位置：光盘\MR\16\419	初级 实用指数：★★★★

实例说明

在使用 JavaScript 语言中的 window 对象的 open()方法打开一个新窗口时，新窗口默认的弹出位置是父窗口的左上方，如果弹出的窗口比较小就不会引起浏览者的注意，这样就达不到预期引起浏览者注意的效果。本实例将实现的是在网页打开时弹出的窗口为居中显示，这样就可以更好地引起浏览者的注意，达到更好的说明宣传效果。本实例运行效果如图 16.3 所示。



图 16.3　弹出窗口居中显示

关键技术

本实例的实现主要是应用了 JavaScript 脚本语言中的 window 对象的 open()方法和 screen 对象，先通过 open()方法打开窗口，然后通过 screen 对象获取屏幕的分辨率，最后通过 window 对象的 moveTo(x,y)方法根据获取的屏幕分辨率的值，将新窗口在父窗口中进行居中显示。下面介绍 window 对象中的 moveTo(x,y)方法和 screen 对象。

使用 moveTo(x,y)方法可以将窗口移到指定的坐标(x,y)的位置。具体语法格式如下：

```
window.moveTo(x,y)
```

JavaScript 语言中的 screen 对象，主要是用来获取当前屏幕的设置的，该对象的常用属性如表 16.1 所示。

表 16.1　screen 对象的常用属性

属　性	说　明
width	用于整个屏幕的水平尺寸，单位是像素
height	用于整个屏幕的垂直尺寸，单位是像素
pixelDepth	显示器每个像素的位数
availWidth	返回的是窗口中内容区域的水平尺寸，单位是像素
availHeight	返回的是窗口中内容区域的垂直尺寸，单位是像素
colorDepth	返回当前颜色的位数设置。1 表示黑白色；8 表示 256 色（支持 256 种颜色）；16 表示增强色（支持 64000 种颜色）；24/32 表示真彩色（支持大概 1600 万种颜色）

设计过程

（1）创建弹出窗口 new.htm。

（2）创建 index.htm 页，编写窗口弹出控制代码，以及控制窗口弹出的位置代码，关键代码如下：

```
<script language="javascript">
        var hdc=window.open('Login_M.htm','','width=322,height=206');
        width=screen.width;
        height=screen.height;
        hdc.moveTo((width-322)/2,(height-206)/2);
</script>
```

秘笈心法

使用 moveTo()方法可以根据需要随意地设置窗口的位置，只要是网站需要的，网站设计者都可以进行相关的设定，以达到预期的效果。

实例 420	通过按钮创建窗口 光盘位置：光盘\MR\16\420	初级 实用指数：★★★★

实例说明

在实际设计网页时，经常需要在一个窗口中创建一个规定了大小、显示位置和显示样式的新窗口，如新用户注册时，进行注册信息的填写。运行本实例，当单击“弹出新窗口”按钮后，将弹出如图 16.4 所示的新窗口。



图 16.4　单击按钮创建的新窗口

关键技术

本实例的实现主要应用了 window 对象中的 open()方法，通过对方法中一些参数的设定，就可以达到预期的效果。open()方法的语法如下：

```
WindowVar=window.open(url,windowname[,location]);
```

参数说明

❶ WindowVar：当前打开窗口的句柄。如果 open()方法成功，则 WindowVar 的值为一个 Window 对象的句柄，否则 WindowVar 的值是一个空值。

❷ url：目标窗口的 URL。如果 URL 是一个空字符串，则浏览器将打开一个空白窗口，允许用 write()方法创建动态 HTML。

❸ windowname：window 对象的名称。

❹ location：对窗口属性进行设置。

设计过程

（1）创建弹出的新窗口 new.htm。

（2）创建网站首页 index.htm，并在首页中编写弹出窗口控制代码，关键代码如下：

```
<body>
<script language="javascript">
function push(){
window.open("new.htm","new","height=400,width=500,top=10,left=20");
}
</script>
<form name="form1" method="post" action="">
  <input type="button" name="Button" value="弹出新窗口" onClick="push()">
</form>
</body>
```

秘笈心法

根据本实例的实现方法，读者可以实现在应用 window 对象的 open()方法打开窗口时，根据不同参数的设置（如窗口的大小、窗口的位置、是否显示工具栏以及状态栏等），使打开的窗口具有不同的效果。

实例 421　**为弹出的窗口加入关闭按钮**　**初级**

光盘位置：光盘\MR\16\421　　实用指数：★★★★

实例说明

在网站中应用弹出窗口时，虽然弹出窗口上有可以直接关闭窗口的按钮，但总是显得不那么直观，为了方便用户，在设计网页时可以直接给弹出窗口添加一个直观的关闭按钮。运行本实例，当单击“最新服务”导航后，将弹出一个带有关闭按钮的窗口，如图 16.5 所示，单击“关闭”按钮窗口将会关闭。



图 16.5　添加关闭按钮

关键技术

本实例的实现主要是应用了 window 对象中的 close()方法，前面在介绍 window 对象时提到过 close()方法，下面将作出详细介绍。

```
window.close();
```

功能：该方法用于自动关闭浏览器窗口。

但是在应用时要注意，应用 close()方法和单击 IE 标题栏中的“关闭”按钮是一样的，但是如果关闭 IE 的主窗口，系统就会弹出“是否关闭此窗口”的确认对话框，只有单击“是”按钮后才可以关闭 IE 主窗口。

设计过程

创建弹出窗口 new.htm，并编写关闭控制语句，关键代码如下：

```
<table width="322" height="206"   border="0" cellpadding="-2" cellspacing="-2">
  <tr>
    <td height="200"   background="001.jpg"><input name="Button" type="button"class="btn_grey" value="关闭"
      onClick="window.close();">
      </td>
  </tr>
```

秘笈心法

读者在设计网站时可以考虑将本实例的实现方法和实例 420 的方法结合，将窗口的弹出做到浏览者可以根据自己的喜好进行控制，这样就会有更好的用户体验。

实例 422	定时打开窗口 光盘位置：光盘\MR\16\422	初级 实用指数：★★★☆

实例说明

弹出窗口有多种方式，有时为了需要，在弹出窗口时要与用户打开窗口之间有一定的时间间隔。本实例就是实现了这一效果，读者在打开网页时，5 秒钟之后才会在网页中弹出新的窗口，这样就给浏览者争取了一部分进行其他操作的时间。本实例运行效果如图 16.6 所示。



图 16.6　定时打开窗口

关键技术

实现本实例的主要方法是 setTimeout()方法和一个自定义的函数，在函数中可以直接调用 window 对象的 open()

方法。对于 setTimeout()方法和 open()方法，请参见实例 418 和实例 420 的关键技术部分。

设计过程

（1）创建弹出窗口 new.htm。

（2）创建网站首页 index.htm，并编写控制弹出窗口的代码，关键代码如下：

```
<script language="javascript">
<!--
function push(){
window.open("new.htm","new","height=190,width=775,top=30,left=30");
}
setTimeout("push()",3000);
-->
</script>
```

秘笈心法

在设计网站时可以结合本实例给出的几种方法，将弹出窗口设计成读者可自行创建和关闭，也可以定时创建手动关闭等效果。

实例 423　关闭弹出窗口时刷新父窗口

光盘位置：光盘\MR\16\423

高级

实用指数：★★★☆

实例说明

用户在浏览网页时都希望网页的信息是最新的，为了达到这一点，在关闭新的弹出窗口时可以对其父窗口进行刷新，也就是用户通过 open()方法打开一个新窗口，在新窗口中对一些数据进行了修改，关闭该子窗口后希望可以在父窗口中看到更新的内容。本实例便实现了这一效果，如图 16.7 所示。



图 16.7　关闭弹出窗口时刷新父窗口

关键技术

本实例主要应用 window.open()语句打开新窗口，并在新窗口中应用 opener 属性，该属性返回一个引用，用于指定打开本窗口的窗口对象。

语法：

```
window.opener
window.opener.方法
window.opener.属性
```

功能：返回的是一个窗口对象。opener 属性与打开该窗口的父窗口相联系，当访问子窗口中的 opener 属性时，返回的是父窗口，通过该属性，可以使用父窗口对象中的方法和属性。

实现本实例还要用到 reload()方法，该方法的功能和浏览器上的刷新按钮是一样的，只要在自定义函数中调用它即可。

设计过程

（1）创建父窗口 index.htm，并做出快捷方式，控制子窗口的弹出。

（2）创建子窗口 new.htm，并编写控制子窗口关闭和刷新父窗口的代码，关键代码如下：

```
<script language="javascript">
function Mycheck()
{
alert("关闭子窗口！");
window.opener.location.reload();        //刷新父窗口中的网页
window.close();                          //关闭当前窗口
}
```

秘笈心法

本实例的实现有着很大的实用价值，即可以让用户更好地浏览到最新的网页信息，获取更大的信息量，用户体验好了，网站的访问量就不是问题了。

实例 424 关闭窗口时不弹出询问对话框

初级

光盘位置：光盘\MR\16\424

实用指数：★★★☆

实例说明

通常情况下，在使用 window 对象中的 close()方法关闭 IE 主窗口时，都会弹出一个“你查看的网页正在试图关闭窗口，是否关闭此窗口？”的询问对话框，这样用户总是要先关闭对话框才可以关闭主窗口，这是很麻烦的。本实例将介绍如何屏蔽对话框，以解决这个问题，其运行效果如图 16.8 所示。



图 16.8　屏蔽对话框效果

关键技术

本实例应用的实现方法就是实例 423 中介绍的 window 对象的 opener 属性，只要将该属性关闭的 IE 窗口的打开窗口设置为 null，这样再使用 window 对象的 close()方法关闭 IE 主窗口时就不会弹出对话框了。

设计过程

创建用于实现效果的网页 index.htm，编写控制代码，关键代码如下：

```
<td width="17%"><a href="#" onClick="window.location.reload();">•  刷新页面</a> <br>
      <a href="#" onClick="window.opener=null;window.close();">•     关闭系统</a>
</td>
```

秘笈心法

本实例的实现方法可以应用到很多网站中，尤其是一些链接比较多的网站，运用此功能可以很大程度上节约用户的浏览时间，省去不必要的麻烦，使用户拥有更好的体验。

实例 425　弹出窗口的 Cookie 控制

光盘位置：光盘\MR\16\425

初级

实用指数：★★★☆

实例说明

很多网站都选择了在用户打开网页时弹出一些新窗口，如广告、网站协议等，如果每次打开网页都会弹出这些新窗口，时间久了会让用户产生厌烦心理，最终会影响到网站的访问量。而本实例的实现就解决了这一问题，它只在用户第一次登录网站时才弹出新窗口。本实例的运行效果如图 16.9 所示，第一次访问时将弹出窗口，当关闭浏览器再次访问时不会弹出窗口。



图 16.9　弹出窗口的 Cookie 控制

关键技术

Cookie 是一个文本文件，是网站在访问者硬盘上存储的一些定制的信息段。通过浏览器，网页可以实现对 Cookie 的存储、获取和删除。Cookie 的目的只有一个，即记录访问者的个体信息。在开始使用 Cookie 前，读者需要知道下面的规则：

浏览器可以存储的总 Cookie 数量不能超过 300 个，每个服务器不得超过 20 个（对于整个服务器，而不仅是用户自己的网页或网站）。存储容量也限制在每个 Cookie 4KB，所以不要试图在一个 Cookie 中存储过多的信息。默认情况下，一个 Cookie 可以在整个浏览器的运行期间存在；当用户退出浏览器后，Cookie 内容就会消失。为了让一个 Cookie 的持续时间超过一个浏览周期，可以设置失效日期。

设计过程

（1）创建弹出窗口 new.htm。

（2）创建控制窗口 index.htm，并在其中编写控制代码，关键代码如下：

```
function openWindow(){
     window.open("new.htm","","width=352,height=193")
}
function GetCookie(name){                //获取登录信息
 var search = name + "=";
 var returnvalue = "";
 var offset,end;
 if(document.cookie.length>0){
      offset = document.cookie.indexOf(search);
```

```
        if(offset != -1){
            offset += search.length;
            end = document.cookie.indexOf(";",offset);
            if(end == -1) end = document.cookie.length;
            returnvalue = unescape(document.cookie.substring(offset,end));
        }
    }
    return returnvalue;
}
function LoadPop(){                    //判断是否是第一次登录，如果是则弹出窗口
    if(GetCookie("pop")==""){
        openWindow();
    }
}
</Script>
```

秘笈心法

如果想再次弹出窗口，可以在浏览器中选择“工具”→“Internet 选项”命令，在弹出的对话框的“常规”选项卡中单击“删除 Cookies”按钮，然后单击“确定”按钮即可。

16.2　弹出网页对话框

本节介绍的网页对话框是指通过指定的脚本代码打开的窗口，这些窗口都是有返回值的。网页对话框可以分为两种，一种是网页模式对话框，另一种是网页非模式对话框。下面将通过具体实例介绍如何应用网页对话框。

实例 426	弹出网页模式对话框 光盘位置：光盘\MR\16\426	初级 实用指数：★★★☆

实例说明

网页对话框的基本操作就是弹出网页模式对话框，只有处理好了弹出网页模式对话框，接下来才好做出其他更多的操作。本实例便实现了弹出网页模式对话框，并对对话框的大小进行了设置。实例运行效果如图 16.10 所示。

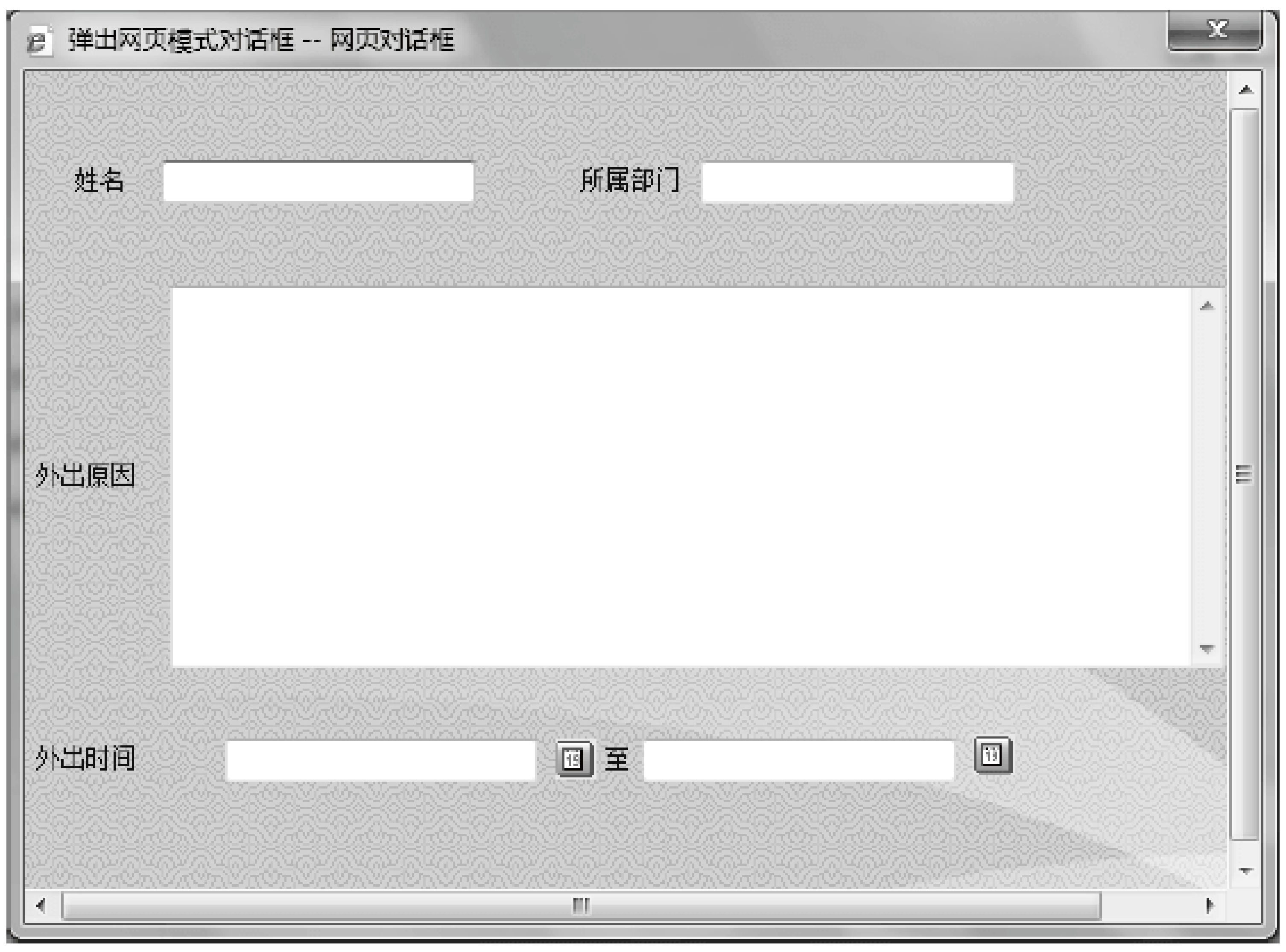


图 16.10　弹出网页模式对话框

关键技术

本实例主要应用 window 对象的 showModalDialog()方法，该方法用于弹出网页（模式）对话框，其语法格

式如下：

```
variant = object.showModalDialog(sURL [, vArguments [, sFeatures]])
```

参数说明

❶ sURL：指定 URL 文件地址。

❷ vArguments：用于向网页对话框传递参数，传递参数的类型不限制，对于字符串类型，最大为 4096 个字符，但也可以传递对象，如 index.htm。

❸ sFeatures：可选参数，窗口对话框的设置参数，所有参数都是可选参数。主要参数如表 16.2 所示。

表 16.2　sFeatures 主要参数

参　　数	说　　明
dialogWidth:number	设置对话框的宽度
dialogHeight:number	设置对话框的高度
dialogTop:number	设置对话框窗口相对于桌面左上角的 Top 位置
dialogLeft:number	设置对话框窗口相对于桌面左侧的 Left 位置
center:{yes | no | 1 | 0 }	指定是否将对话框在桌面上居中，yes|1 为居中显示，no|0 为不居中显示，默认值为 yes
help: {yes|no 1|0}	指定对话框窗口中是否显示上下文敏感的帮助图标。默认值是 yes
scroll: {yes|no 1|0}	指定对话框是否出现滚动条
resizable:{yes|no|1|0}	指定对话框窗口大小是否可变，默认值是 no
status: {yes|no 1|0}	指定对话框是否显示状态栏

■ 设计过程

（1）创建弹出的网页模式对话框 new.htm。

（2）创建控制页面 index.htm，并在控制页面编写控制代码，关键代码如下：

```
<script language="javascript">
function opendialog()
{
var someValue=window.showModalDialog("new.htm","","dialogWidth=640px;dialogHeight=423px;status=no;help=no;scrollbars=no")
}
</script>
```

■ 秘笈心法

本实例的实现内容可以应用到很多地方，如商业网站的注册信息的填写，以及一些需要随时更新的网站公告等。

实例 427　全屏显示网页模式对话框

光盘位置：光盘\MR\16\427

初级

实用指数：★★★☆

■ 实例说明

在进行网站开发时，有时一些信息只有特定的用户才能看到，而且必须看到，那么在设计时就可以将这些信息放到一个网页模式对话框中，在进行显示时实现全屏显示即可。本实例便实现了该功能，在用户单击相关超链接以后，弹出相应的网站服务规定和信息，运行效果如图 16.11 所示。

■ 关键技术

本实例主要应用 screen 对象的 width、height 属性和 window 对象的 showModalDialog()方法实现，showModalDialog()方法的具体介绍请参见实例 426。打开网页对话框除了使用 showModalDialog()方法外，还可

使用 showModelessDialog()方法。



图 16.11　全屏显示网页模式对话框

showModalDialog()和 showModelessDialog()方法两者的主要区别就在于 showModalDialog()打开的网页对话框是模式窗口，并且是置在父窗口上的，要想关闭父窗口，就必须先关闭该对话框；而 showModelessDialog()方法打开的对话框是无模式窗口，打开后无论关闭与否都可以访问父窗口或其他窗口。

设计过程

（1）创建全屏弹出窗口 new.htm。

（2）创建控制弹出窗口的主窗口 index.htm，并将控制代码写入，关键代码如下：

```
<script language="javascript">
function opendialog()
{
        var width=screen.width;
        var height=screen.height;
        window.showModalDialog("new.htm","","dialogWidth="+width+"px;dialogHeight="+height +"px;status=no;help=no;scrollbars=no")
}
</script>
<td width="64"><a href="#" onClick="opendialog()">购买须知</a></td>
```

秘笈心法

实现本实例后，再设计网站时，就可以做到强制用户去注意一些网站的规定，这样可以更好地约束用户的行为，维护网站的正常合法运营。

实例 428	实现网页日期选择 光盘位置：光盘\MR\16\428	初级 实用指数：★★★★

实例说明

在进行网页人机交互时，经常会遇到一些网页需要进行时间的输入，如果让用户手工输入的话，那么在时间的格式上是很难控制的，而且还不方便操作。为了解决该问题，可以在网页中加入日期选择器，让用户自行选择设计好格式的时间。本实例便实现了这一效果，运行效果如图 16.12 所示。



图 16.12　实现网页日期选择

关键技术

本实例主要应用 window 对象的 showModalDialog()方法实现，showModalDialog( )方法的详细介绍请读者参见实例 426。

设计过程

（1）创建网站首页 index.htm，在本页创建弹出窗口网页控制快捷方式。

（2）创建需要进行日期选择的网页 new.htm，编写控制函数 loadCalendar()，具体代码如下：

```
<script language="javascript">
function loadCalendar(field)
{
    var rtn = window.showModalDialog("calendar.htm","","dialogWidth:280px;dialogHeight:250px;status:no;help:no;scrolling=no;scrollbars=no");
if(rtn!=null)
        field.value=rtn;
        return;
}
</script>
```

（3）创建日期选择页面 calendar.htm，并编写日期显示代码，本部分代码是实现本实例的关键部分，关键代码如下：

```
<script language=javascript>
    var monthNames = new Array ( "", "1","2","3","4","5","6","7", "8", "9", "10", "11", "12" );
    var endDay = new Array ( 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 );
    var dayNow = 0;
    var monthNow = 0;
    var yearNow = 0;
    function load ( form ) {
        set_month_year_now ();
        var found = false;
        for ( var month=0; month<form.monthList.length; month++ )
            if ( form.monthList[month].text == monthNames[monthNow] ) {
                form.monthList[month].selected = true;
                found = true;
            }
        if ( !found) {
            error ();
            return;
        }
        var found = false;
        for ( var year=0; year<form.yearList.length; year++ )
             if ( form.yearList[year].text == yearNow ) {
                form.yearList[year].selected = true;
                found = true;
             }
        if ( !found) {
            error ();
```

```
            return;
        }
        display_month ( form );
    }
function preceding_month ( form ) {
        var month_selected = form.monthList.selectedIndex;
        var year_selected = form.yearList.selectedIndex;
        if ( !month_selected && !year_selected )    {
            error ();
            return;
        }
        if ( month_selected > 0 )
            month_selected --;
        else {
            month_selected = 11;
            year_selected --;
        }
        form.monthList[month_selected].selected = true;
        form.yearList[year_selected].selected = true;
        display_month ( form );
    }
    function following_month ( form ) {
        var month_selected = form.monthList.selectedIndex;
        var year_selected = form.yearList.selectedIndex;
        if ( month_selected >= ( form.monthList.length - 1 ) && year_selected >= ( form.yearList.length - 1 ) ) {
            error ();
            return;
        }

        if ( month_selected < 11 )
            month_selected ++;
        else {
            month_selected = 0;
            year_selected ++;
        }
        form.monthList[month_selected].selected = true;
        form.yearList[year_selected].selected = true;
        display_month ( form );
    }
    function set_month_year_now ()    {
        var form = document.calendar;
        var now = new Date ();
        monthNow = now.getMonth () + 1;
        yearNow = now.getYear ();
        dayNow = now.getDate();
        yearNow = ( yearNow < 100 ) ? yearNow + 1900 : yearNow;
        var count = 0
        for (var i = yearNow-103; i < yearNow + 50; i++) {
          eval("form.yearList.options[count] = new Option('"+i+"', '"+i+"')");
          count++;
        }
        form.yearList.selectedIndex = 103;
        form.yearList.length = count;
    }
    function error ()    {
        alert ( "超出范围!" );
    }
    function display_month ( form )    {
        var month = form.monthList.selectedIndex + 1;
        var year = parseInt ( form.yearList.options[ form.yearList.selectedIndex].text );
        var start_day = start_day_in_month ( year, month );
        var count = 0;
        for ( var row=0; row<6; row++) {
            for ( var col=0; col<7; col++ )
            {
                if ( row == 0 && col < ( start_day - 1 ) )
                    var day = "";
                else if ( count < endDay[month] )
                    day = ++count;
```

```
                    else
                        day = "";
                    form.dayBox[(row*7)+col].style.display = "";
                    form.dayBox[(row*7)+col].style.color = "black";
                    if (day == "") {
                      form.dayBox[(row*7)+col].style.display = "none";
                    } else {
                     form.dayBox[(row*7)+col].value = day;
                     if (col%7 == 0) form.dayBox[(row*7)+col].style.color = "red";
                     if (yearNow == year && monthNow == month && dayNow == day) form.dayBox[(row*7)+col].style.color = "blue";;
                  }
                }
            }
        }
        function start_day_in_month ( year, month )     {
            var    day, daynum, ndays, mnum;
            sday = start_day_in_year ( year );
            endDay[2] = ( year % 4 ) ? 28 : 29;
            if ( month == 1 )
                daynum = sday;
            else {
                ndays = sday;
                for ( mnum=2; mnum<month+1; mnum++ )
                    ndays = ndays + endDay[mnum-1];
                daynum = ndays % 7;
            }
            daynum = (!daynum) ? 7 : daynum;
            return (daynum);
        }
        function start_day_in_year ( year )     {
            var y, m, d;
            var n;
            y = year - 1; m = 13; d = 1;
            n = d + 2 * m + ( Math.floor ( ( 0.6 + (m + 1) ) ) + y );
            n = n + Math.floor ( ((y / 4) - Math.floor ( (y / 100 ) ) + Math.floor ( ( y / 400 ) ) )        ) + 2 ;
            n = Math.floor ( ( (n / 7 - Math.floor ( (n / 7) ) ) * 7 + 0.5 ) );
            return (n+1);
        }
      function CheckDate(strDay) {
       var docFrm = document.calendar;
       var choice_daynum = 0;
       var current_daynum = 0;
       var day_temp;
       if (strDay != "") {
          var strY = docFrm.yearList.value;
          var strM = docFrm.monthList.value;
          var curr_y = new String(yearNow);
          var curr_m = new String(monthNow);
          var curr_d = new String(dayNow);
          if (curr_m.length == 1) curr_m = "0"+curr_m;
          if (curr_d.length == 1) curr_d = "0"+curr_d;
          current_daynum = new Number(curr_y + curr_m + curr_d) ;
          if (strM.length == 1) strM = "0"+strM;
          if (strDay.length == 1) strDay = "0"+strDay;
             choice_daynum = new Number(strY + strM + strDay);
             parent.window.returnValue = strY+"-"+strM+"-"+strDay;    //将选择的日期传递到父窗口中
             parent.window.close();
       }
       return false;
      }
</script>
<body onLoad="load(document.calendar)" topmargin="0">
<center>
   <form name="calendar">
      <table border="0" cellpadding="0" cellspacing="0">
         <tr>
            <td colspan="3" height="24"></td>
         </tr>
         <tr>
```

```
            <td width="205" align="right" VALIGN="MIDDLE" nowrap="nowrap">
        <select name="yearList" onChange="display_month(this.form)">
            </select></td>
            <td width="65" nowrap="nowrap" align="left"><select name="monthList" size="1" onChange="display_month(this.form)">
                    <option value="1">1</option>
                    <option value="2">2</option>
                    <option value="3">3</option>
                    <option value="4">4</option>
                    <option value="5">5</option>
                    <option value="6">6</option>
                    <option value="7">7</option>
                    <option value="8">8</option>
                    <option value="9">9</option>
                    <option value="10">10</option>
                    <option value="11">11</option>
                    <option value="12">12</option>
                </select>
            </td>
            <td width="10"></td>
        </tr>
        <tr>
            <td colspan="3" height="6"></td>
        </tr>
        <tr>
            <td colspan="3"><table border="0" cellpadding="1" cellspacing="0" align="center">
                    <tr>
                        <td bgcolor="#82664F"><table border="0" cellpadding="0" cellspacing="0">
                            <tr bgcolor="#82664F" height="18">
                                <td width="31" align="center" nowrap="nowrap" BGCOLOR="#0099FF"><font color="#FF0000">日</font></td>
                                <td width="31" align="center" nowrap="nowrap" BGCOLOR="#0099FF">一</td>
                                <td width="31" align="center" nowrap="nowrap" BGCOLOR="#0099FF">二</td>
                                <td width="31" align="center" nowrap="nowrap" BGCOLOR="#0099FF">三</td>
                                <td width="31" align="center" nowrap="nowrap" BGCOLOR="#0099FF">四</td>
                                <td width="31" align="center" nowrap="nowrap" BGCOLOR="#0099FF">五</td>
                                <td width="31" align="center" nowrap="nowrap" BGCOLOR="#0099FF">六</td>
                            </tr>
                            <tr bgcolor="#FFFFFF" height="18">
                                <td align="center"><input type="text" size="2" name="dayBox" readOnly onClick="javascript:CheckDate(this.value);"
onMouseOver="this.style.background='#EEEEEE'" onmouseout="this.style.background='white'">
                                </td>
                                <td align="center"><input type="text" size="2" name="dayBox" readOnly onClick="javascript:CheckDate(this.value);"
onMouseOver="this.style.background='#EEEEEE'" onmouseout="this.style.background='white'">
                                </td>
                                <td align="center"><input type="text" size="2" name="dayBox" readOnly onClick="javascript:CheckDate(this.value);"
onMouseOver="this.style.background='#EEEEEE'" onmouseout="this.style.background='white'">
                                </td>
                                <td align="center"><input type="text" size="2" name="dayBox" readOnly onClick="javascript:CheckDate(this.value);"
onMouseOver="this.style.background='#EEEEEE'" onmouseout="this.style.background='white'">
                                </td>
                                <td align="center"><input type="text" size="2" name="dayBox" readOnly onClick="javascript:CheckDate(this.value);"
onMouseOver="this.style.background='#EEEEEE'" onmouseout="this.style.background='white'">
……
                                </td>
                                <td align="center"><input type="text" size="2" name="dayBox" readOnly onClick="javascript:CheckDate(this.value);"
                                </td>

                            </tr>

                        </table></td>
                    </tr>
                </table></td>
        </tr>
    </table>
  </form>
</center>
</body>
</html>
```

秘笈心法

其实在设计网页时，不仅时间可以让用户进行选择，性别、爱好等也都可以进行选择输入，这样不仅可以方便用户操作，而且可以控制输入内容的格式。

实例 429　**网页拾色器**　光盘位置：光盘\MR\16\429　初级　实用指数：★★★★

实例说明

在一些网站中，尤其一些博客论坛中总会有浏览者留言这个模块。为了突出留言者的个性和引人注意，在留言时可以将字体颜色进行相应的改变。本实例便实现了在进行留言编辑时用户可以选择自己喜欢的字体颜色，运行效果如图 16.13 所示。

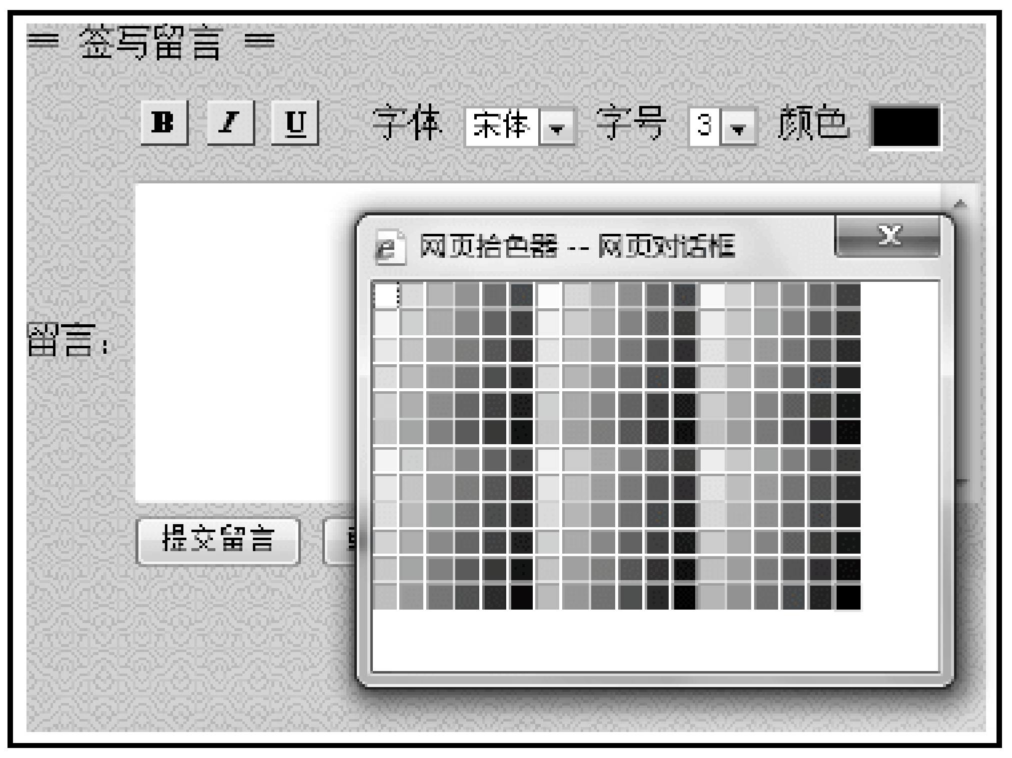


图 16.13　网页拾色器效果

关键技术

在本实例中仅使用 216 种浏览器安全的颜色，即所谓 Netscape 色块。这 216 种颜色分别代表 0、51、102、153 和 204 这 5 个颜色值以及每一种原色（即红、绿、蓝）。这些十进制数值对应的十六进制数分别为 0x00、0x33、0x66、0x99、0xCC 和 0xFF。在 HTML 的颜色属性中，黑色是#000000，纯红色是#FF0000，纯绿色是#00FF00，纯蓝色是#0000FF，白色是#FFFFFF。在实现网页拾色器时，需要应用 JavaScript 的数组。创建数组可以有以下 3 种方法。

（1）无参数调用。语法格式如下：

```
var h = new Array();
```

（2）指定数组前 n 个元素的值。语法格式如下：

```
var h = new Array(arglist);
```

其中参数 arglist 是一个用逗号隔开的值表，这些值用于给 Variant 所包含的数组的各元素赋值。如果不提供参数，则创建一个长度为 0 的数组。

（3）指定具有的元素个数。语法格式如下：

```
var h = new Array(n);
```

其中参数 n 是指定数组的长度。由于在 JavaScript 中，数组的第一个元素的下标值为 0，所以 n 的值为数组的最大下标值加 1。

设计过程

（1）创建 index.htm 网页，并在其中编写控制代码，实现调用拾色器页面，关键代码如下：

```
<script language="javascript">
function colorpick(field){
    var rtn = window.showModalDialog("color.htm","","dialogWidth:225px;dialogHeight:170px;status:no;help:no;scrolling=no;scrollbars=no");
    if(rtn!=null)
```

```
        field.style.background=rtn;
        return;
    }
</script> 颜色
<input name="color" type="text" id="color" size="3" readonly="yes" style="background-color:#000000" onClick="colorpick(this);">
</span></td>
```

（2）创建拾色器页面 color.htm，并编写相应的自定义函数，关键代码如下：

```
<body>
<script language="JavaScript">
<!--
var h = new Array(6)
h[0] = "FF";
h[1] = "CC";
h[2] = "99";
h[3] = "66";
h[4] = "33";
h[5] = "00";
function action(RGB) {
  //alert('您选择的颜色是:#'+RGB);
  parent.window.returnValue="#"+RGB;
  window.close();
}
function Mcell(R, G, B) {
  document.write('<td bgcolor="#' + R + G + B + '">');
  document.write('<a href="#" onClick="action(\'' + (R + G + B) + '\')">');
  document.write('<img border=0 height=12 width=12 \')" alt=\'#'+R+G+B+'\'>');
  document.write('</a>');
  document.write('</td>');
}
function Mtr(R, B) {
  document.write('<tr>');
  for (var i = 0; i < 6; ++i) {
    Mcell(R, h[i], B);
  }
  document.write('</tr>')
}
function Mtable(B) {
  document.write('<table cellpadding=0 cellspacing=0 border=0>');
  for (var i = 0; i < 6; ++i) {
    Mtr(h[i], B);
  }
document.write('</table>');
}
function Mcube() {
  document.write('<table cellpadding=0 cellspacing=0 border=0><tr>');   //
  for (var i = 0; i < 6; ++i) {
      if(i%3==0){
      document.write('<tr>');
      }
    document.write('<td bgcolor="#FFFFFF">');
    Mtable(h[i])
    document.write('</td>');
  }
      if(i%3==0){
      document.write('</tr>');
      }
  document.write('</tr></table>');
}
Mcube()
  -->
</script>
</body>
```

秘笈心法

根据本实例，读者可以实现在开发网站时，使用网页拾色器实现网页的换肤功能。

16.3　窗口的动画效果

在浏览网页时，许多窗口在打开时都显示了各种各样的动画效果，这样可以在浏览页面时，给人一种动态、新颖的效果。下面将通过几个典型实例进行详细讲解。

实例 430　**页面自动滚动**　　初级

光盘位置：光盘\MR\16\430　　实用指数：★★★★

实例说明

本实例主要实现了在用户打开网页时，网页自动纵向滚动，显示网页底部，用户只有在调整窗口到合适大小时才可看到全部网页内容。本实例还实现了标题的横向循环滚动。运行效果如图 16.14 所示。



图 16.14　页面自动滚动

关键技术

本实例主要是应用 window 对象的 scroll()方法指定窗口的当前位置。下面对 scroll()方法的使用进行详细的介绍。

scroll()方法的语法格式如下：

```
scroll(x,y);
```

参数说明

❶ x：屏幕的横向坐标。

❷ y：屏幕的纵向坐标。

功能：指定窗口显示位置的坐标。

设计过程

创建操作页面 index.htm，并在其中使用 scroll()方法实现效果，主要代码如下：

```
<script language="JavaScript">
var position = 0;
function scroller(){
    if (true){
        position++;
        scroll(0,position);
        clearTimeout(timer);
        var timer = setTimeout("scroller()",10);
    }
}
scroller();
</script>
```

秘笈心法

对于本实例的实现方式，读者可以将其运用到一些网页中，这样用户在浏览网页时就必须查看全部网页内容，从而可以更好地获取网页中的全部信息，包括一些网页规范，有利于网站的正常运营。

实例 431　动态显示网页

光盘位置：光盘\MR\16\431

初级　实用指数：★★★★

实例说明

本实例主要实现了在用户打开网页时，所弹出的窗口是最小状态，然后窗口的高度慢慢增加，直到和屏幕的高度一样，当高度和屏幕一样后，窗口的宽度便开始增长，直到和屏幕的宽度一样，这时整个窗口便稳定下来不再变化。实例运行效果如图 16.15 所示。



图 16.15　动态显示网页

关键技术

本实例主要应用了 screen 对象的 availWidth 和 availHeight 属性来获得当前屏幕的宽度和高度，并用 resizeTo()方法来自动增加窗口的高度和宽度。screen 对象的介绍请参见实例 419 的关键技术部分。下面将详细介绍 resizeTo()方法，其具体格式如下：

```
window.resizeTo(x,y)
```

参数说明

❶ x：窗口的水平宽度。

❷ y：窗口的垂直宽度。

功能：将窗口改变成(x,y)大小，x、y 分别为宽度和高度。

设计过程

创建动态显示的窗口 index.htm，在其中调用 window 对象的 resizeTo()方法，关键代码如下：

```
<script language="JavaScript">
function expandingWindow() {
 var heightspeed = 2;
 var widthspeed = 7;
 var leftdist = 0;
 var topdist = 0;
 var winwidth = window.screen.availWidth - leftdist;
 var winheight = window.screen.availHeight - topdist;
 for (sizeheight = 1; sizeheight < winheight; sizeheight += heightspeed) {
```

```
        self.resizeTo("1", sizeheight);
}
for (sizewidth = 1; sizewidth < winwidth; sizewidth += widthspeed) {
        self.resizeTo(sizewidth, sizeheight);
}
}
</script>
```

秘笈心法

根据本实例的实现方法，读者可以试着去做到限定窗口的打开大小，可以制作下拉式的窗口，从而实现更好的网页动态效果。

实例 432　指定窗口的扩展大小

光盘位置：光盘\MR\16\432

初级

实用指数：★★★★

实例说明

本实例的实现效果是当窗口弹出时，单击窗口中的超链接，会在屏幕的左上角弹出子窗口，并且弹出的子窗口的高度和宽度会慢慢变大，直到和整个屏幕一样，窗口才会稳定不再变化。本实例的运行效果如图 16.16 所示。



图 16.16　指定窗口的扩展大小

关键技术

本实例主要是应用 window 对象的 open()方法来打开窗口，用 screen 对象的 availHeight 和 availWidth 属性来获取屏幕可工作区域，用 moveTo()和 resizeTo()方法来设置窗口的位置和大小，用 resizeBy()方法使窗口逐渐变大，直到和屏幕的大小相同。下面主要对 window 对象的 resizeBy()方法进行介绍。

resizeBy()方法的语法格式如下：

```
window.resizeBy(x,y)
```

参数说明

❶ x：窗口放大或缩小的水平宽度。

❷ y：窗口放大或缩小的垂直宽度。

功能：改变窗口的大小为(x,y)，当 x、y 的值大于 0 时为扩大，小于 0 时为缩小。

设计过程

（1）创建指定大小的弹出窗口 new.htm。

（2）创建控制窗口 index.htm，并编写控制高宽变化的函数 goh1()和 gow2()，关键代码如下：

```
<script language=JavaScript>
var winheight,winsize,x;
function goh1(){
```

```
winheight=100;
winsize=100;
x=5;
win2=window.open("new.htm","","scrollbars='no'");
win2.moveTo(0,0);
win2.resizeTo(100,100);
gow2();
}
function gow2(){
if (winheight>=screen.availHeight-3)
        x=0
win2.resizeBy(5,x)
winheight+=5
winsize+=5
if (winsize>=screen.width-5){
        winheight=100
        winsize=100
        x=5
        return
}
setTimeout("gow2()",50)
}
</script>
```

秘笈心法

本实例中 resizeBy()方法的应用可以加以扩展，不仅可以设定为全屏显示，还可以将其设定为只在屏幕的左上角的一部分区域进行显示，读者可以试验一下。

实例 433　实现空降窗口

光盘位置：光盘\MR\16\433

高级

实用指数：★★★★

实例说明

网页设计时的窗口弹出方式很多，本实例便实现窗口从屏幕的上方直接空降下来。实例运行效果如图 16.17 所示。



图 16.17　实现空降窗口

关键技术

本实例的实现首先是用 screen 对象的 availHeight 属性获取屏幕的工作区的高度，然后使用 window 对象的 moveBy()方法实现窗口的下降操作。moveBy()方法的语法如下：

```
window.moveBy(x,y)
```

参数说明

❶ x：水平位移量。

❷ y：垂直位移量。

功能：按照指定位移量设置窗口的大小。

设计过程

创建下降动态页面 index.htm，在其中自定义函数 down()中调用 moveBy()方法，实现实例效果，关键代码如下：

```
<script>
function down() {
      var heig=window.screen.availHeight;
 if(self.moveBy){
         self.moveBy (0,-heig);
         for(i = Math.floor(heig/3); i > 0; i--){
                  self.moveBy(0,3);
         }
 }
}
</script>
```

秘笈心法

根据本实例的实现方法，读者还可以实现让窗口斜着弹出等动态效果。

实例 434	慢慢变大窗口 光盘位置：光盘\MR\16\434	初级 实用指数：★★★★

实例说明

本实例主要实现的效果是在弹出窗口时先指定窗口的弹出位置，然后让窗口慢慢变大，直到和屏幕的工作区大小一样为止。实例运行效果如图 16.18 所示。

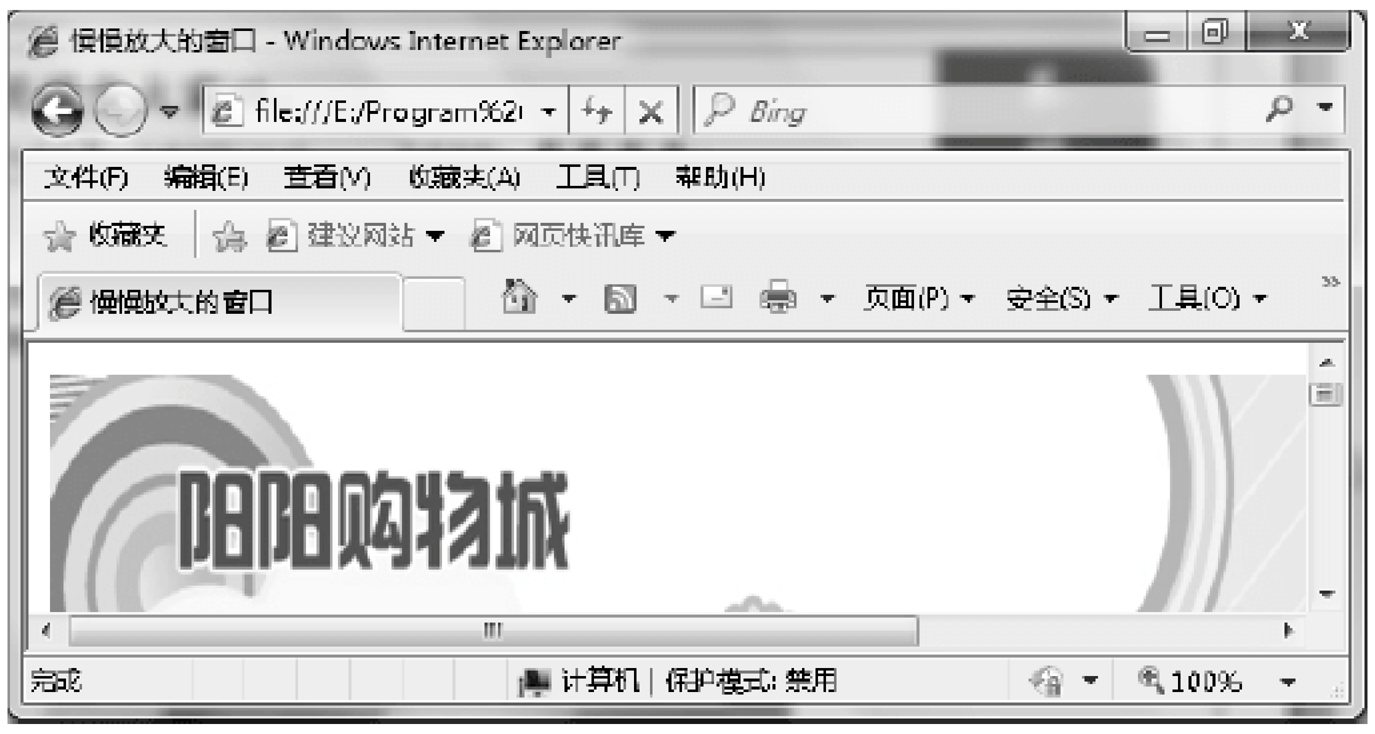


图 16.18　慢慢变大窗口

关键技术

本实例的实现主要是应用了 screen 对象的 availHeight 和 availWidth 属性获取屏幕的工作区的大小，然后应用 window 对象的 resizeTo()方法指定窗口的弹出位置，用 moveTo()方法改变窗口的大小。

设计过程

创建变化窗口 index.htm，在其中编写自定义函数 bbig()，并在函数中调用 window 对象的 resizeTo()和 moveTo()方法实现实例效果，关键代码如下：

```
<BODY onLoad="bbig()">
<table width="800" height="476" border="0" align="center" cellpadding="0" cellspacing="0">
```

```
  <tr>
    <td width="800"><img src="12.jpg" width="800" height="800" /></td>
  </tr>
</table>
<script language="JavaScript">
var sun=800;
var screenW=screen.availWidth;
var screenH=screen.availHeight;
window.resizeTo((screenW-sun),(screenH-sun));
var initialW=screenW-sun;
var initialH=screenH-sun;
var x=((screenW)-initialW)/2;
var y=((screenH)-initialH)/2;
window.moveTo(x,y);
var timer;
function bbig(){
 if (initialH<(window.screen.height-30)){
        initialW=initialW+2;
        initialH=initialH+2;
        window.resizeTo(initialW,initialH);
        timer=setTimeout("bbig()",50);
        x=x-1;
        y=y-1;
        window.moveTo(x,y);
 }
 else{window.clearTimeout(timer)}
}
</script>
</BODY>
```

秘笈心法

在本实例的实现基础上，读者可以试着让窗口慢慢变小和在屏幕的右下角进行窗口向左上角慢慢放大，也可以试着让窗口进行缩小。

实例 435	移动的窗口 光盘位置：光盘\MR\16\435	初级 实用指数：★★★★

实例说明

很多读者都见过泡泡的屏保，泡泡在屏幕中来回地弹跳很好看，本实例便实现了窗口从屏幕的左上角弹出，然后在屏幕中来回移动，当窗口碰到屏幕边缘时就弹开，直到关闭窗口，移动的效果才会停止。实例运行效果如图 16.19 所示。



图 16.19　移动的窗口

关键技术

本实例主要应用 screen 对象的 availHeight 和 availwidth 两个属性获取屏幕的工作区，然后判断窗口是否碰到屏幕边缘，如果碰到就弹开。还应用了 window 对象的 resizeTo()和 moveTo()方法进行指定窗口的大小和位置。

设计过程

创建移动窗口 index.htm，在其中自定义函数 ltir()，在函数中使用 screen 对象以及 window 对象的方法 resizeTo()和 moveTo()，就可以实现本实例的效果，关键代码如下：

```
<body>
<img src="12.jpg" width="782" height="600">
<script language="JavaScript">
window.resizeTo(300,300)
window.moveTo(0,0)
inter=setInterval("ltor()", 1);
var aa=0
var bb=0
var a=0
var b=0
function ltor()
{
try{
if (aa==0)
a=a+2;
if (a>screen.availWidth-300)
aa=1;
if (aa==1)
a=a-2;
if (a==0)
aa=0;
if (bb==0)
b=b+2;
if (b>screen.availHeight-300)
bb=1;
if (bb==1)
b=b-2;
if (b==0)
bb=0;
window.moveTo(a,b);
}
catch(e){}
}
</script>
</body>
```

秘笈心法

读者可以根据本实例的实现方法让网页中出现一些移动的装饰，也可以将网站的一些协议和注意事项在窗口中移动出现，从而更好地引起用户的注意。

实例 436	震颤窗口 光盘位置：光盘\MR\16\436	初级 实用指数：★★★★

实例说明

用过 QQ 等聊天工具的读者都知道，其中有一个颤屏功能，每一次颤屏都会引起对方注意。本实例就是实现窗口的颤屏功能，实例的运行效果如图 16.20 所示，当单击窗口中的相应按钮时，窗口将颤动。



图 16.20　震颤窗口

关键技术

本实例主要使用了 window 对象的 moveBy()方法，方法的详细介绍请参见实例 433 的关键技术部分，此处不再赘述。

设计过程

创建操作网页 index.htm，在其中编写相关的自定义函数，在函数中调用 window 对象的 moveBy()方法，实现实例效果，关键代码如下：

```
<body>
<p>
    <SCRIPT language=javascript>
function flap_xy(n){
 if (self.moveBy){
        for (i = 10; i > 0; i--){
            for (j = n; j > 0; j--){
                self.moveBy(0,i);
                self.moveBy(i,0);
                self.moveBy(0,-i);
                self.moveBy(-i,0);
            }
        }
}
}
function flap_x(n){
 if (self.moveBy){
        for (i = 10; i > 0; i--){
            for (j = n; j > 0; j--){
                self.moveBy(i,0);
                self.moveBy(-i,0);
            }
        }
}
}
function flap_y(n){
 if (self.moveBy){
        for (i = 10; i > 0; i--){
            for (j = n; j > 0; j--){
                self.moveBy(0,i);
                self.moveBy(0,-i);
            }
        }
}
}
   </SCRIPT>
    <input name="button1" type="button" id="button1" value="左右震动" onclick="flap_x(5)">
    <input name="button2" type="button" id="button2" value="上下震动" onclick="flap_y(5)">
    <input name="button3" type="button" id="button3" value="整屏震动" onclick="flap_xy(5)">
</p>
```

```
<p><img src="11.jpg" width="779" height="502"></p>
</body>
```

秘笈心法

moveBy()方法的应用还有很多，不同的参数设定有不同的效果。读者可以试着改变其中的参数，看看是否可以得到更多的效果。

实例 437　旋转的窗口

光盘位置：光盘\MR\16\437

初级

实用指数：★★★★

实例说明

本实例实现的是让窗口在屏幕中以一点为圆心进行旋转，让用户更好地注意到该窗口，这样可以在窗口中加一些广告信息等，达到更好的宣传效果。本实例运行效果如图 16.21 所示。

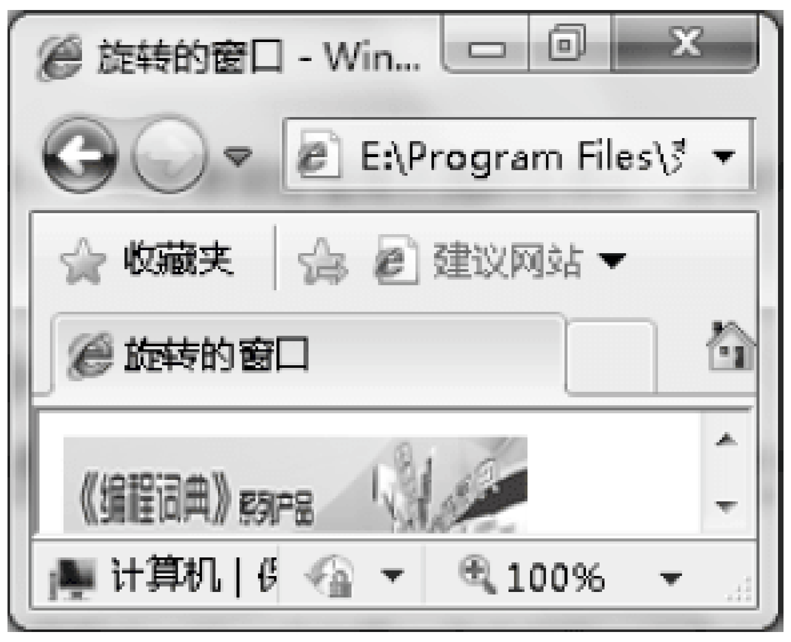


图 16.21　旋转的窗口

关键技术

实现本实例的关键问题是如何计算出窗口运动时的圆形轨迹，这里可以用公式 r*Math.cos((i*Math.PI)/180)和 r*Math.sin((i*Math.PI)/180)计算得出，这样得出坐标值后就可以使用 window 对象的 moveTo()方法实现窗口的旋转。两个公式的参数说明如下。

- r：指的是圆形轨迹的半径。
- i：指的是圆周的点，通过它可以求出圆轨迹的各点坐标。

设计过程

创建实例操作窗口 index.htm，编写自定义函数 eddy()，在自定义函数中调用 window 对象的 moveTo()方法和圆形轨迹的计算公式，具体代码如下：

```
<body>
<img align="left" src="001.jpg" width="150" height="60">
<script language="javascript">
var a=275;
var b=275;
var pp;
var r=60;
var x=y=0;
window.resizeTo(200,200);
function eddy(i){
 var ob=document.all("tt");
 x = r*Math.cos((i*Math.PI)/180)+a;
 y = r*Math.sin((i*Math.PI)/180)+b;
 window.moveTo(x,y);
 i=i+1;
 if (r>100){window.clearTimeout(pp);}
 else{
        if (i>360){i=0;}
                pp=setTimeout("eddy("+i+")",10);
 }
}
eddy(0);
</script>
</body>
```

秘笈心法

本实例的实现中要计算出圆形的轨迹，这需要很好的数学知识。所以作为一名网站设计和制作者一定要对

各方面知识都有所了解，这样才能在网页设计时得心应手。

16.4 窗 口 控 制

窗口的控制在网站制作和网页设计中所占的分量是很重的。例如，有时要控制网页的大小不变、控制窗口的最大化和最小化、根据屏幕的分辨率调整窗口的大小等。下面就通过一些实例来详细介绍。

实例 438 始终将窗口居上显示

光盘位置：光盘\MR\16\438

高级

实用指数：★★★☆

实例说明

在设计网站时经常需要强制用户对某些信息进行处理，例如，有些网站内容需要进行注册后才可以浏览。本实例就实现了将窗口居上显示，只有用户处理好本窗口的内容才可以操作其父窗口或其他窗口的内容。本实例运行效果如图 16.22 所示。



图 16.22 将窗口居上显示

关键技术

本实例主要应用了 document 对象的 focus()方法来获得窗口的焦点。用 window 对象的 open()方法进行窗口操作，将其返回值即窗口对象赋给一个变量，然后用该变量对创建的窗口进行操作。下面将对 open()方法进行详细介绍。

window 对象的 open()方法的语法格式如下：

```
WindowVar=window.open(url,windowname[,location]);
```

参数说明

❶ url：用于打开一个新的浏览器窗口，并在新窗口中装入一个指定的 URL 地址。

❷ windowname：打开一个新的浏览器窗口时，指定窗口的名称。

❸ location：对窗口属性进行设置，其可选参数如表 16.3 所示。

表 16.3 location 参数的取值

参 数 值	描 述
toolbar	指定窗口是否有标准工具栏。当该选项的值为 1 或 yes 时，表示有标准工具栏；当该选项的值为 0 或 no 时，表示没有标准工具栏
location	指定窗口是否有地址工具栏，选项的值及含义与 toolbar 相同
directories	指定窗口是否有链接工具栏，选项的值及含义与 toolbar 相同
status	指定窗口是否有状态栏，选项的值及含义与 toolbar 相同
menubar	指定窗口是否有菜单，选项的值及含义与 toolbar 相同
scrollbar	指定当前窗口文档大于窗口时是否有滚动条，选项的值及含义与 toolbar 相同
resizable	指定窗口是否可改变大小，选项的值及含义与 toolbar 相同
alwaysLowered	指定窗口隐藏在所有窗口之后，选项的值及含义与 toolbar 相同
alwaysRaised	指定窗口浮在所有窗口之上，选项的值及含义与 toolbar 相同
dependent	指定打开的窗口为当前窗口的一个子窗口，并随着父窗口的关闭而关闭，选项的值及含义与 toolbar 相同
hotkeys	在没有菜单栏的新窗口中设置安全退出的热键，选项的值及含义与 toolbar 相同

续表

参　数　值	描　　述
innerHeight	设定窗口中文档的像素高度
innerWidth	设定窗口中文档的像素宽度
screenX	设定窗口距离屏幕左边界的像素长度
screenY	设定窗口距离屏幕上边界的像素长度
titleBar	指明标题栏是否在新窗口中可见，选项的值及含义与 toolbar 相同
z-look	指明当窗口被激活时，不能浮在其他窗口之上，选项的值及含义与 toolbar 相同

设计过程

创建效果启动页面 index.htm，在其中利用 JavaScript 脚本语言的相关对象和函数输出一个新的弹出窗口让其始终居上显示，关键代码如下：

```
<body>
<input type="button" name="Button" value="显示窗口" onClick="newform()">
<script language="JavaScript">
var name;
function newform(){
 name=window.open("","","width=300,height=200");
 name.document.write('<input name="imageField" type="image" src="001.jpg" width="280" height="180"
border="0">');
 show(name);
}
function show(name){
 try{
 name.document.focus();
 setTimeout("show(name)",1);
 }
 catch(e){}
}
</script>
```

秘笈心法

本实例实现的方法可以强制用户进行必要的信息处理，其实强制用户进行信息处理的方法很多，如可以在用户不处理相应的信息时不让其进行登录或在本网站中进行网页的跳转等。

实例 439	窗口全屏显示 光盘位置：光盘\MR\16\439	高级 实用指数：★★★☆

实例说明

用户在浏览网页时经常需要对一些窗口进行全屏显示，以便更清晰地浏览网页中的信息。窗口在弹出时若不是全屏显示的话，通常都是用户自行操作进行全屏，如果只是应用窗口自身的全屏按钮总是不那么方便，那么在网页设计时就可以给网页加上全屏按钮。本实例就是实现这一功能的，运行效果如图 16.23 所示。

关键技术

本实例主要应用了 window 对象的 open()方法中的 fullscreen 参数和 document 对象的 location 属性。使用 open()方法中的 fullscreen 参数将当前窗口重新创建，这样既可以对本窗口进行全屏显示，又可以避免其他窗口在打开时也全屏显示。而 document 对象的 location 属性则是用来指定当前窗口。



图 16.23　全屏显示窗口

设计过程

（1）创建初始窗口 index.htm，并在其中添加快捷方式，关键代码如下：

```
<div align="center">
<input name="button1" type="button" id="button1" onClick="fullscreen()" value="全屏显示">
<input name="button2" type="button" id="button2" value="关闭窗口" onClick="window.close()">
</div>
```

（2）编写 JavaScript 脚本语言，实现新窗口的弹出和控制，关键代码如下：

```
<script language="javascript">
var pp;
function fullscreen(){
 pp=window.open("Index.htm","",'fullscreen=yes');
}
</script>
```

秘笈心法

细心的读者可以发现，好的网站的一些操作都是很人性化的，它们都做到了方便用户的操作，这样就会有很好的用户体验，拥有了好的用户体验何愁访问量的问题呢。

实例 440	自动最大化窗口 光盘位置：光盘\MR\16\440	高级 实用指数：★★★☆

实例说明

实例 439 中使用按钮进行窗口的最大化，其实在设计网页时可以对一些信息量大或者信息比较重要的网页直接实现自动最大化窗口。本实例就实现了这一功能，运行效果如图 16.24 所示。

关键技术

本实例主要是用 screen 对象的 availWidth 和 availHeight 属性来获取屏幕工作区的宽度和高度，并用 resizeTo()方法将当前窗口的大小设为屏幕大小，再用 moveTo()方法将窗口放在屏幕的左上角。



图 16.24　自动最大化窗口

设计过程

创建弹出窗口 index.htm，在其中嵌入最大化控制代码，使用 window 对象的 moveTo()和 resizeTo()两个方法，关键代码如下：

```
<BODY onLoad="clockon()">
<table width="800" height="476" border="0" align="center" cellpadding="0" cellspacing="0">
  <tr>
    <td width="800"><img src="01.jpg" width="800" height="600" /></td>
  </tr>
</table>
<script language="JavaScript">
this.moveTo(0,0)
this.resizeTo(screen.availWidth,screen.availHeight)
</script>
</BODY>
```

秘笈心法

修改实例中的 resizeTo()方法的参数可以实现窗口任意指定大小。

实例 441	按钮实现最大和最小化 光盘位置：光盘\MR\16\441	高级 实用指数：★★★☆

实例说明

平时用户在浏览网页时，经常会遇到一些弹出的窗口的大小不适合自己的需要，有时需要窗口最大化，有时又需要窗口最小化。这些操作用户都可以使用窗口本身的最大和最小化按钮去实现，但是由于本身所带的按钮使用起来很不方便，所以在进行网站设计时可以在网页的显示窗口添加最大化和最小化按钮。本实例就是实现这一功能的，实例运行效果如图 16.25 所示。

关键技术

本实例主要应用到了<object>标记和<param>标记。<object>标记中的 classid 属性是给出浏览器插件的类型。用<object>标记在页面中插入 ActiveX 控件或其他对象之后，有时需要向该对象或者控件传递参数，这就要使用<param>标记。该标记没有相应的结束标志</param>，下面对<param>标记进行详细说明。



图 16.25　最大和最小化按钮实现

<param>标记的一般格式为：

```
<param name=* value=* valuetype=* type=*>
```

参数说明

❶ name：是参数的名字。

❷ value：指定参数的值。

❸ valuetype：指定怎样表示参数的值。

❹ type：指定媒体类型。

设计过程

创建网页 index.htm，并在其中写入相关属性参数和 JavaScript 代码，关键代码如下：

```
<script language="JavaScript">
var s="";
s=s+'<object id=hh1 classid="clsid:ADB880A6-D8FF-11CF-9377-00AA003B7A11">';
s=s+'<param name="Command" value="Minimize"></object>';
s=s+'<object id=hh2 classid="clsid:ADB880A6-D8FF-11CF-9377-00AA003B7A11">';
s=s+'<param name="Command" value="Maximize"></object>';
aa.innerHTML=s;
</script>
```

秘笈心法

良好的操作界面是衡量一个网站好坏的重要标准，读者在设计网站时一定要注意如何把握网页中的快捷方式的设计，尽量提供简洁方便的操作。

实例 442	频道方式的窗口 光盘位置：光盘\MR\16\442	高级 实用指数：★★★★☆

实例说明

在浏览网页时，经常会看到一些网页以频道的方式进行打开。本实例就是实现这一功能的，用户打开网页后单击其中的“频道方式”按钮，即会弹出窗口的频道方式的显示模式。本实例运行效果如图 16.26 所示。

关键技术

本实例主要应用了 window 对象的 open()方法中的 channelmode 参数值来使弹出的窗口以频道方式进行显示。



图 16.26 频道方式显示窗口

设计过程

（1）创建频道弹出窗口 new.htm。

（2）创建频道弹出窗口的控制窗口 index.htm，并在其中调用 window 对象的 open()方法和设定方法的相关参数，关键代码如下：

```
<body>
<script language="javascript">
function show(){
        window.open("new.htm","频道方式窗体","channelmode,scrollbars");
}
</script>
<form name="form1" method="post" action="">
  <input type="button" name="Submit" value="频道方式窗口" onClick="show()">
</form>
</body>
```

秘笈心法

读者可以在实现本实例的基础上，根据 open()方法参数的说明，试着去改变一些参数的值，使窗口的显示更加多样化。

实例 443	根据用户分辨率自动调整窗口 光盘位置：光盘\MR\16\443	高级 实用指数：★★★☆

实例说明

在浏览网页时，给浏览者以清晰、整齐的感觉是进行网站设计时最大的追求。但有时可能会因为浏览者计算机的分辨率不同，而影响页面的整体效果。本实例将实现根据不同计算机分辨率来自动调整窗口，实例运行效果如图 16.27 所示。

关键技术

本实例主要应用 screen 对象获取屏幕的分辨率，然后用 window 对象的 resizeTo()方法设置窗口的大小。在前面的实例中已经讲解了 screen 对象和 window 对象的 resizeTo()方法的具体用法，此处不再详细说明。



图 16.27　自动调整的窗口

设计过程

创建显示窗口 index.htm，在其中编写自定义函数 winsize()，在该函数中调用 window 对象的 resizeTo()方法，实现本实例的效果，关键代码如下：

```
<script language="javascript">
function winsize()
{
window.resizeTo(screen.width-100,screen.height-100);
}
</script>
</head>
<body onload="winsize();">
<table width="800" height="476" border="0" align="center" cellpadding="0" cellspacing="0">
  <tr>
    <td width="800"><img src="11.jpg" width="800" height="600" /></td>
  </tr>
</table>
</body>
```

秘笈心法

由于不同版本的浏览器或用户的硬件设置的不同都可能影响到网页的显示效果，所以在进行网页设计时以上因素都要考虑到，尽量减少固定参数的设置。

实例 444	使窗口背景透明 光盘位置：光盘\MR\16\444	高级 实用指数：★★★☆

实例说明

在浏览网页时，常常需要在网页中进行一些文本的输入，但网页的背景常常会使输入的内容模糊不清。本实例就是实现在文本输入框中实现背景透明的，实例运行效果如图 16.28 所示。



图 16.28　窗口背景透明显示

关键技术

本实例主要应用的是浮动框架标记<iframe>。用<iframe>标记可以将其他页中的内容显示在当前页面中。下面对<iframe>标记的相关属性进行说明。<iframe>标记的属性如表 16.4 所示。

表 16.4　<iframe>标记的属性

属　　性	说　　明
src	浮动框架中显示页面源文件的路径
width	浮动框架的宽度
height	浮动框架的高度
name	浮动框架的名称
align	浮动框架的排列方式，left 表示居左，center 表示居中，right 表示居右
frameborder	设置框架边框显示
scrolling	设置框架滚动条显示
marginwidth	设置框架边缘宽度
marginheight	设置框架边缘高度
allowtransparency	设置框架背景透明

设计过程

（1）创建输入窗口 melody.htm。

（2）创建控制窗口 index.htm，并在其中编写控制代码和设定相关参数，关键代码如下：

```
<table valign="center" align="center" width="480" height="318" border="1" cellpadding="0" cellspacing="0">
      <tr>
        <th width="480" height="318" valign="middle" scope="col">
          <div id="div1" align="justify"></div>
      </th>
      </tr>
</table>
<script language="JavaScript">
div1.innerHTML='<IFRAME frameborder="0" scrolling="no" src="melody.htm" width=506 height=360
allowtransparency></iframe>';
</script>
```

秘笈心法

本实例的实现方式还可以应用到如网站内容的清晰显示、动态地替换输入信息等操作。读者在设计网页时一定要注意网页的视觉效果。

16.5　框架的应用

所谓框架就是网页的各部分为相互独立的网页，又由一个网页将这些分开的网页组成一个完整的网页，显示在浏览者的浏览器中，重复出现的内容被固定下来，每次浏览者发出对页面的请求时，只下载发生变化的框架页面，其他子页面保持不变。使用框架可以将容器窗口划分为若干个子窗口，在每个子窗口中可以分别显示不同的网页。下面将通过具体实例详细介绍框架的应用。

实例 445	框架集的嵌套 光盘位置：光盘\MR\16\445	高级 实用指数：★★★☆

实例说明

在进行网络程序开发时，在网页中使用一个框架集是非常简单的。但如果要在同一个网页中应用更多的框架集时，应该怎样进行框架集的嵌套呢？本实例将解决这一问题，运行结果如图 16.29 所示。

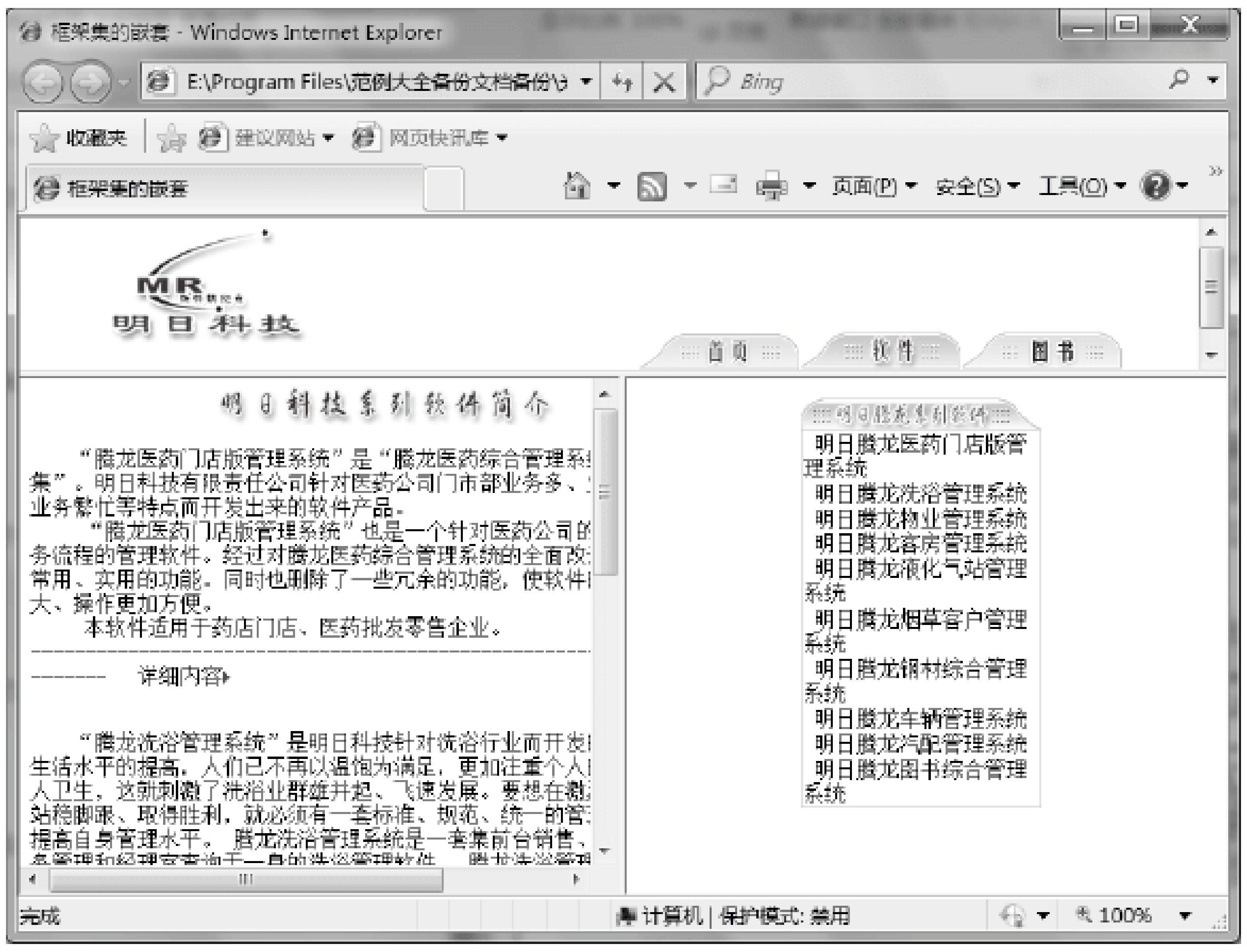


图 16.29　框架集的嵌套

关键技术

本实例主要介绍框架集的嵌套应用，下面将对框架集进行详细介绍。

框架集主要包含如何组织各个框架的信息，可以通过<frameset>标记来定义框架集。框架是按照行和列来组织的，可以使用<frameset>标记的下列属性对框架的结构进行设置。

（1）左右分割窗口属性 COLS

在水平方向上将浏览器分割成多个窗口，可以通过框架的左右分割窗口属性 COLS 实现，语法格式如下：

```
<frameset cols="value,value,……">
  <frame>
  <frame>
</frameset>
```

参数说明

value：用于指定各个框架的列宽，取值有 3 种形式，即像素、百分比（%）和相对尺寸（*）。

例如，若要通过框架将浏览器窗口划分为 3 列，其中第一列占浏览器窗口宽度的 20%，第二列为 120 像素，第 3 列为浏览器窗口的剩余部分的框架，关键代码如下：

```
<frameset cols="20%,120,*" >
  <frame>
  <frame>
</frameset>
```

（2）上下分割窗口属性 ROWS

在垂直方向上将浏览器分割成多个窗口，可以通过框架的上下分割窗口属性 ROWS 实现，语法格式如下：

```
<frameset rows="value,value,……">
  <frame>
  <frame>
</frameset>
```

参数说明

value：用于指定各个框架的行高，取值有 3 种形式，即像素、百分比（%）和相对尺寸（*），设置方法与 COLS 属性类似。

例如，若要通过框架将浏览器窗口划分为 3 行，其中的第一行占浏览器窗口宽度的 20%，第二行为 120 像素，第 3 行为浏览器窗口的剩余部分的框架，关键代码如下：

```
<frameset rows="20%,120,*" >
  <frame>
  <frame>
</frameset>
```

（3）框架边框显示属性 FRAMEBORDER

该属性用于指定框架周围是否显示边框，取值为 1（显示边框，默认值）或 0（不显示边框）。

（4）FRAMESPACING

该属性用于指定框架之间的间隔，以像素为单位。如果不设置该属性，则框架之间没有间隔。

（5）指定边框宽度属性 BORDER

该属性用于指定边框的宽度，只有 FRAMEBORDER 属性为 1 时才有效。

设计过程

在需要进行布局的页面中写入以下程序代码，将 src 属性值修改为需要链接的文件即可，关键代码如下：

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>框架集的嵌套</title>
</head>
<!--创建嵌套框架集-->
<frameset rows="30%,*">
    <frame src="software_top.htm">
    <frameset cols="50%,50%">
    <frame src="software_main.htm">
    <frame src="software_left.htm" scrolling="auto">
    </frameset>
</frameset>
<body>
</body>
</html>
```

秘笈心法

框架的合理应用可以使网页的显示更加合理和模块化，使网站的维护更加容易。

实例 446　在网页中应用浮动框架

光盘位置：光盘\MR\16\446

高级

实用指数：★★★☆

实例说明

在开发网络程序时，经常会应用到框架，如网络考试管理系统中就涉及了浮动框架。通过浮动框架将网站中各部分独立的网页重新组成一个完整的网页，并显示在浏览器中。运行本实例，在网页的左侧是一个独立的网页；右侧也是一个独立的网页，如图 16.30 所示。

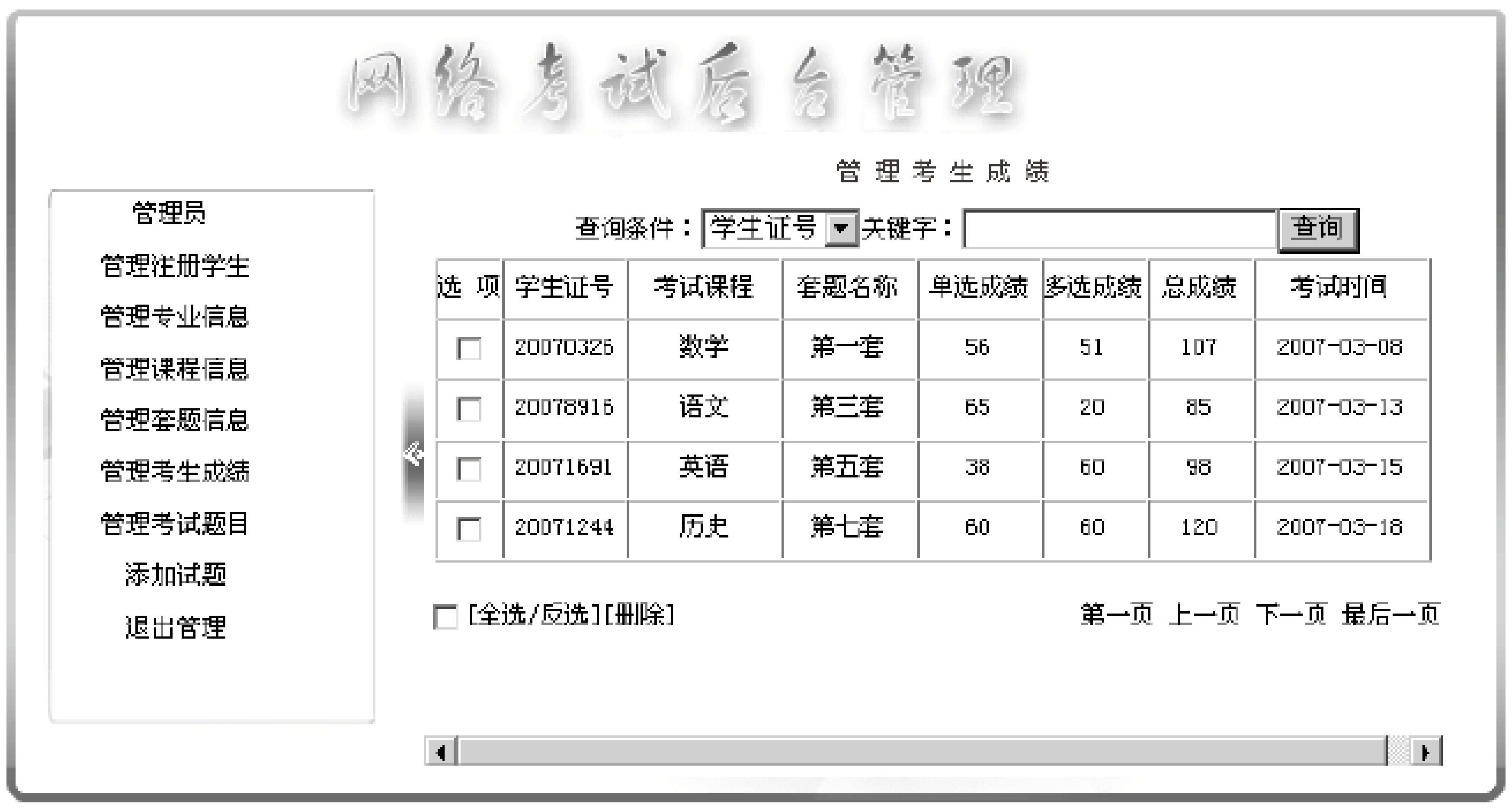


图 16.30　在网页中应用浮动框架

关键技术

本实例主要应用浮动框架将各部分独立的网页重新组成一个完整的网页，下面将对浮动框架进行详细介绍。

浮动框架<iframe>是一种特殊的框架页面，在浏览器窗口中可以嵌套子窗口，在其中显示子页面的内容，语法格式如下：

```
<IFRAME src="文件" height="数值" width="数值" name="框架名称" scrolling="值" frameborder="值">
</IFRAME >
```

下面是浮动框架属性的详细介绍。

（1）浮动框架的文件路径属性 SRC

语法格式如下：

```
<IFRAME SRC="file_name">
```

参数说明

file_name：指明浮动框架文件的文件名或者其他超链接的网址。

（2）浮动框架的名称属性 NAME

语法格式如下：

```
<IFRAME SRC="file_name" NAME="frame_name">
```

参数说明

frame_name：定义的浮动框架名称。

（3）浮动框架的对齐属性 ALIGN

语法格式如下：

```
<IFRAME SRC="file_name" ALIGN="left/center/right">
```

参数说明

❶ left：居左对齐。

❷ center：居中对齐。

❸ right：居右对齐。

（4）浮动框架的宽度和高度属性 WIDTH、HEIGHT

语法格式如下：

```
<IFRAME SRC="file_name" WIDTH="value" HEIGHT="value">
```

参数说明

❶ WIDTH：浮动框架的宽度。

❷ HEIGHT：浮动框架的高度。

（5）浮动框架滚动条显示属性 SCROLLING

语法格式如下：

```
<IFRAME SRC="file_name" SCROLLING="value">
```

参数说明

value：有 3 个取值，分别为 YES（显示滚动条）、NO（不显示滚动条）、 AUTO（根据窗口内容决定是否有滚动条）。

（6）浮动框架边框属性 FRAMEBORDER

语法格式如下：

```
<IFRAME SRC="file_name" FRAMEBORDER="value">
```

参数说明

value：值为 yes 代表显示框架边框，值为 no 代表隐藏框架边框。

（7）浮动框架边缘的宽度和高度属性 MARGINWIDTH、MARGINHEIGHT

语法格式如下：

```
<IFRAME SRC="file_name" MARGINWIDTH="value" MARGINHEIGHT="value">
```

参数说明

❶ MARGINWIDTH：设定浮动框架左右边缘与边框的宽度。

❷ MARGINHEIGHT：设定浮动框架上下边缘与边框的高度。

设计过程

在需要重新组建的网页中写入浮动框架的程序代码，并将 src 属性值设为需要组建的网页文件，关键代码如下：

```
<table cellpadding="0" cellspacing="0" border="0" width="100%" height="100%">
<tr>
<td width="6%"></td>
<td  height="100%"><iframe  src="adm_Mainleft.htm"  frameborder="0"  width="100%"  height="100%"  name="mainl"  align=  "middle" scrolling="auto">对不起，你的浏览器不支持框架！</iframe></td>
<td width="7%">
</td>
</tr>
</table>
</td></tr>
<tr>
<td height="36"> </td>
</tr>
</table>
</td>
<td width="1%">
<img src="images/Manage/gotoleft.gif" alt="放大"   width=12 height="100">
</td>
<td  width="490" height="300">
<iframe src="adm_Mainright.htm" frameborder="0" width="100%" height="100%" name="mainr" scrolling="auto">对不起，你的浏览器不支持框架！
</iframe>
</td>
</tr>
</table>
```

秘笈心法

读者还可将本实例的实现方法应用到后台管理的网页中，这样就可以省去网页跳转时带来的麻烦。

实例 447	创建空白框架 光盘位置：光盘\MR\16\447	高级 实用指数：★★★☆

实例说明

在进行网络程序开发时常常需要应用到框架，所谓框架就是网页的各部分为相互独立的网页，又由一个网页将这些分开的网页组成一个完整的网页，显示在浏览者的浏览器中，此时重复出现的内容将被固定下来。那么如何在一个网页中创建空白的框架呢？本实例将解决这一问题，实例运行结果如图 16.31 所示。



图 16.31　创建空白框架

关键技术

本实例主要应用框架的 src 属性给框架写一个通用的 HTML 文档。下面将对 Frame 框架进行详细介绍。

使用 Frame 框架可以设置框架的属性，包括框架的名称、框架是否包含滚动条以及在框架中显示的网页等，语法格式如下：

```
<Frame src="文件" name="框架名称" scrolling="值" [noresize] frameborder="数值">
```

下面对框架属性进行详细介绍。

（1）框架文件的路径属性 SRC

功能：通过 SRC 属性来定义框架装载文件的路径。

语法格式如下：

```
<FRAME SRC="file_name">
```

参数说明

file_name：指明框架文件的文件名或者其他超链接的网址。

（2）框架的名称属性 NAME

功能：通过 NAME 属性定义框架的名称，所起的名称将应用于页面的链接和脚本描述。

语法格式如下：

```
<FRAME SRC="file_name" NAME="frame_name">
```

（3）框架边框显示属性 FRAMEBORDER

功能：可以设置是否显示框架边框。框架边框的显示情况继承框架集边框属性的设定。

语法格式如下：

```
<FRAME SRC="file_name" FRAMEBORDER="value">
```

参数说明

value：值为 yes 代表显示框架边框，值为 no 代表隐藏框架边框。

（4）框架滚动条显示属性 SCROLLING

功能：通过 SCROLLING 属性设置是否在框架内显示滚动条。

语法格式如下：

```
<FRAME SRC="file_name" SCROLLING="value">
```

参数说明

value：有 3 个取值，分别为 yes（显示滚动条）、no（不显示滚动条）、auto（根据窗口内容决定是否显示滚动条）。

（5）框架边缘的宽度和高度属性 MARGINWIDTH、MARGINHEIGHT

功能：通过设置这两个属性可以调整框架页面内容与边框的距离。

语法格式如下：

```
<FRAME SRC="file_name" MARGINWIDTH="value" MARGINHEIGHT="value">
```

参数说明

❶ MARGINWIDTH：设定框架左右边缘与边框的宽度。

❷ MARGINHEIGHT：设定框架上下边缘与边框的高度。

（6）框架尺寸调整属性 NORESIZE

功能：禁止改变框架的尺寸。利用该属性可以控制框架的尺寸是否可以调整。

语法格式如下：

```
<FRAME SRC="file_name" NORESIZE>
```

（7）不支持框架标记<NOFRAMES>

某些版本的浏览器是不支持框架结构的，如果遇到这种情况，就可以使用<NOFRAMES>和</NOFRAMES>标记再声明一对文件主体标记<BODY>和</BODY>，代表在无法接受框架结构时唯一显示的页面。

设计过程

在需要添加空白框架的页面中添加以下程序代码，即可完成空白框架的创建，关键代码如下：

```
<html>
<head>
<script type="text/javascript">
<!--
function sun()
{
return "<html></html>";
}
//-->
</script>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>创建空白框架</title>
</head>
<frameset>
<frame name="Frame1" src="ss.htm">
```

```
<frame name="Frame2" src="javascript:parent.sun()">
</frameset>
</html>
```

秘笈心法

读者可以和 JavaScript 脚本语言的内容进行结合，在浏览器不支持框架时弹出对话框进行提示。

实例 448	居中显示框架 光盘位置：光盘\MR\16\448	高级 实用指数：★★★☆

实例说明

将页面设置为居中显示对于用表格布局的页面来说很简单，只需设置表格水平居中显示即可，但如果是框架页面就没有那么简单了，这就需要应用框架嵌套技术。例如，在明日科技网站的软件首页就是应用框架嵌套技术居中显示页面的，如图 16.32 所示。

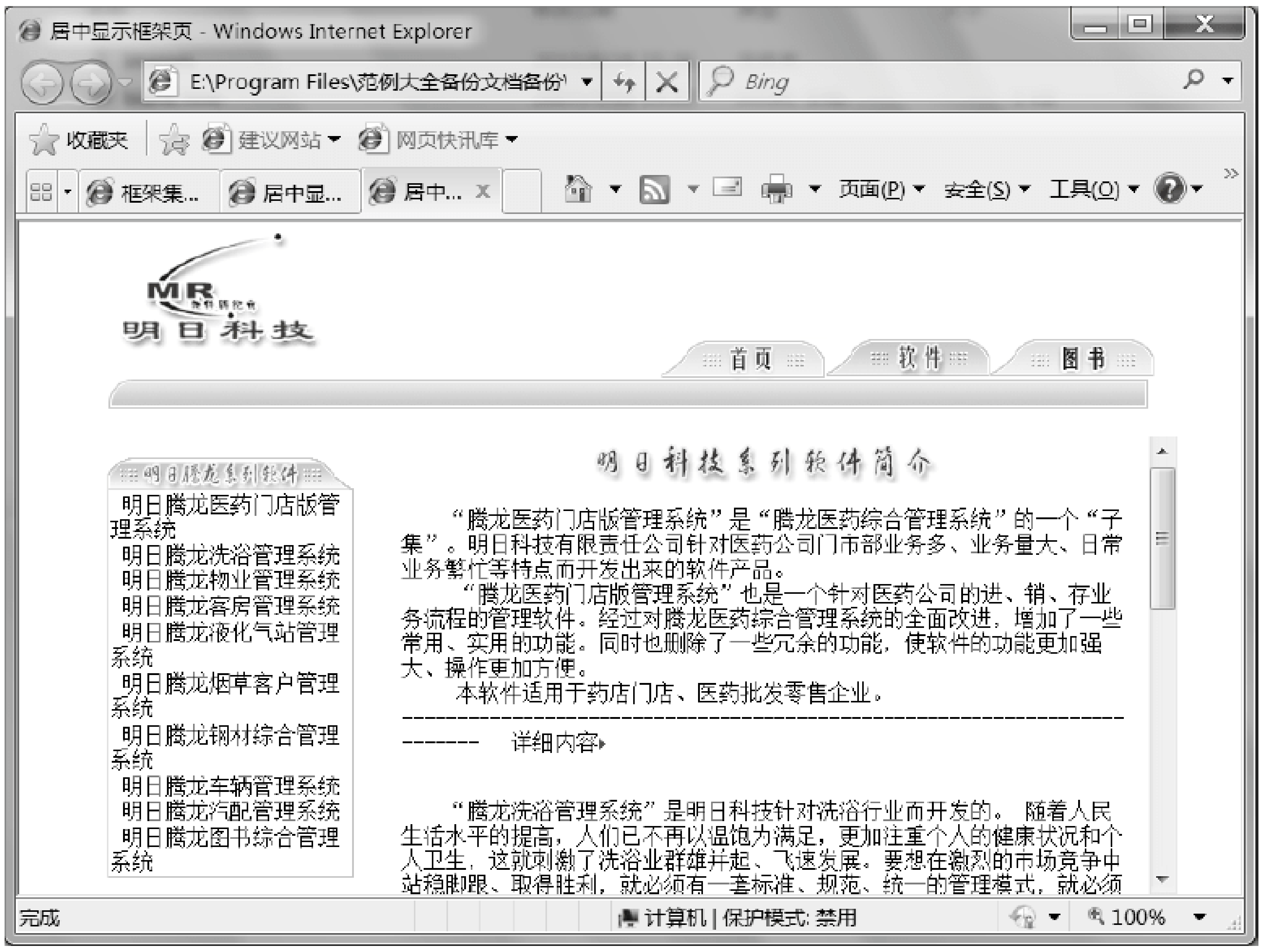


图 16.32　居中显示框架

关键技术

本实例主要应用框架集实现框架页的居中显示，框架集的详细介绍请读者参见实例 445。

设计过程

通过对框架集<frameset>标记的属性设置，将指定的框架页进行居中显示，关键代码如下：

```
<html>
<head>
<title>居中显示框架页</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>
<frameset rows="*" cols="1*,800,1*" frameborder="NO" framespacing="0" border="0">
  <frame src="blank.htm" name="BLFrame" scrolling="NO" noresize>
<frameset rows="157,*" cols="*" frameborder="NO" border="0" framespacing="0">
  <frame src="software_top.htm" name="topFrame" scrolling="NO" noresize >
  <frameset rows="*" cols="225,*" framespacing="0">
```

```
    <frame src="software_left.htm" name="leftFrame" scrolling="auto" noresize>
    <frame src="software_main.htm" name="mainFrame" frameborder="no" scrolling="auto">
  </frameset>
</frameset>
<frame src="blank.htm" name="BRFrame" scrolling="NO" noresize>
</frameset>
</frameset>
<noframes>
<body>
</body>
</noframes>
</html>
```

秘笈心法

在建设 Web 网站时不得不考虑的一个问题是：如何让不同分辨率的用户都能看到网页的最佳效果。目前，虽然许多用户的浏览器的分辨率还是 800×600 的，但也有一部分已经是 1024×768 了，所以在设计页面时，经常要兼顾两者。那么就应该这样设计页面：将网页在 800×600 的分辨率下设计，然后将页面居中显示，这样的网页在 1024×768 下显示时，将不会给人很难看的感觉。

16.6　无边框窗口

所谓无边框窗口，是指不包括 IE 浏览器窗口所固有的标题栏及灰色的窗口边框的窗口，无边框窗口在某些特定的网站中起着举足轻重的作用。下面将通过几个具体实例介绍如何实现无边框窗口。

实例 449	全屏显示无边框有滚动条的窗口 光盘位置：光盘\MR\16\449	高级 实用指数：★★★☆

实例说明

通常用户在 IE 浏览器中浏览网页时，浏览器窗口都是包括标题栏、菜单栏和状态栏等固定内容的。虽然通过 IE 主菜单可以将浏览器的菜单栏和状态栏隐藏，但是标题栏却不能隐藏，即使将网页全屏显示也是不能将其去除。有时为了实现网站的整体效果或其他目的，不得不使用全屏显示的无边框有滚动条的窗口。例如，在皮皮宠物网站中，为了给用户以更真实的感觉，就可以将整个网站的主页面设置为全屏显示模式，如图 16.33 所示。

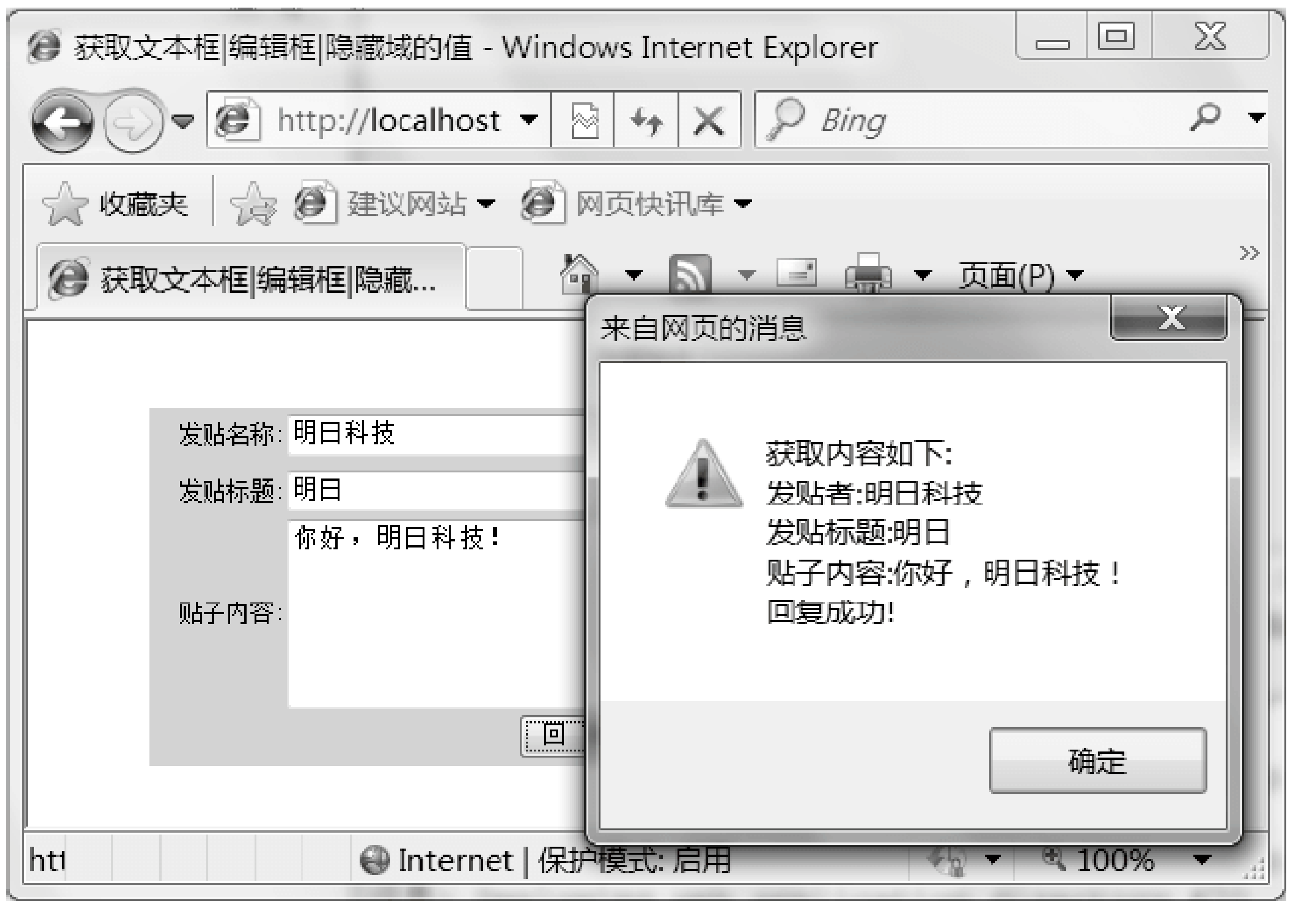


图 16.33　全屏显示无边框有滚动条的窗口

关键技术

实现全屏显示无边框有滚动条的窗口主要通过 window 对象的 open()方法实现。这就需要借助一个中转页实现，即当该中转页运行时调用 open()方法打开一个全屏显示的窗口，并关闭打开中转页的窗口，这时就需要应用 window 对象的 close()方法关闭窗口，关闭窗口时不弹出确认对话框。

设计过程

（1）新建一个空的 HTML 页面，该页面中可以没有任何页面布局元素或内容，名称为 index.htm。在该页面中添加的代码如下：

```
<script language="javascript">
winClose();
function winClose()
{
    window.opener=null;
    window.close();
    window.open("main.htm","","fullscreen=1");
}
</script>
```

（2）创建 main.htm 文件，该文件才是网站的真正首页，前面的 index.htm 页面只是起到一个页面跳转的作用。该页面读者可根据实际情况设计，这里不再进行详细介绍。

秘笈心法

在 JavaScript 中调用 window 对象的 open()方法时，只要设置 fullscreen 参数值为 1，即可将打开的新窗口全屏显示。

实例 450	应用 CSS 实现指定尺寸无边框无滚动条窗口 光盘位置：光盘\MR\16\450	高级 实用指数：★★★☆

实例说明

对于一个页面风格清新自然的网站来说，如果弹出的窗口带有不适合页面风格的灰色边框及死板的标题栏，那么这势必会影响网站的整体效果。本实例将向读者介绍一种更为完善的无边框窗口的实现方法，那就是应用 CSS+DIV 实现。运行本实例，单击“用户登录”超链接即可弹出无边框的用户登录窗口，如图 16.34 所示。



图 16.34　应用 CSS 实现指定尺寸无边框无滚动条窗口

关键技术

本实例主要通过在页面中加入 DIV 层，并在页面中控制层的位置及其显示和隐藏来实现。在页面中加入层的语法如下：

```
<DIV id="value" align="value" class="value" style="value">
    ……
</DIV>
```

参数说明

❶ id：<div>标签的 id 也可以说是它的名字，常与 CSS 样式结合，实现对网页中任何元素的控制。

❷ align：用于控制<div>标签中的元素的对齐方式，其 value 值可以是 left、center 和 right，分别用于设置元素的居左、居中和居右对齐。

❸ class：用于设置<div>标签中的元素的样式，其 value 值为 CSS 样式中的 class 选择符。

❹ style：用于设置<div>标签中的元素的样式，其 value 值为 CSS 属性值，各属性值间应用分号分隔。该属性最常用的功能之一就是进行<div>标签的定位，其对应的属性为 position，该属性有两个可选值，即 relative 和 absolute，relative 表示<div>标签的位置是相对于它所在的窗口的，absolute 表示<div>标签的位置是绝对的，可以通过表 16.5 所示的属性进行设置。

表 16.5　<div>的属性说明

属　　性	说　　明
left	相对于窗口左边的位置，单位为像素（pixels）
top	相对于窗口上边的位置，单位为像素（pixels）
width	<div>标记的宽度。所有在<div>标记中的文字或 HTML 元素都包含在里面
height	<div>标记的高度。该属性很少使用，除非想要对层进行分割
clip:rect(top,right,bottom,left)	给出层的可见部分。该属性可使得<div>标记显示为一个可以定义得很准确的方块区域
visibility	隐藏或显示<div>标记中的元素，其值为 visible、hidden、inherit
z-index	<div>标记的立体位置，值越大，<div>标记的位置越高
background-color	<div>标记的背景颜色
layer-background-color	Netscape 的<div>标记的背景颜色
background-image	<div>标记的背景图像
layer-background-image	Netscape 的<div>标记的背景图像

style 属性的另一个常用功能是控制<div>标记的 display 属性，用于设置元素的浮动特征，当 display 被设置为 block（块）时，容器中所有元素都将会被当作一个单独的块放入到页面中；将 display 设置为 inline，将使其行为和元素 inline 一样，即使是普通的块元素它也会被组合成像<span>那样的输出流输出到页面上；将 display 设置为 none，<div>元素就像从页面中被移走一样，它下面的所有元素都会被自动跟上填充。

设计过程

（1）在要弹出无边框窗口的页面的最底部加入一个 DIV 层，其 name 属性的值为 User，并通过其 style 属性控制层的大小和层的隐藏，该层中的内容为无边框窗口要显示的内容。User 层的完成代码如下：

```
<!--应用层设计用户登录表单页面开始-->
<div id="User" style="position:absolute;width:240px; height:139px;display:none;">
<table width="240" height="139"    border="0" align="center" cellpadding="-2" cellspacing="-2">
  <tr>
    <td height="139" align="center"><form name="form_U" method="post" action="#">
        <table width="220" height="115" bgcolor="#FFFFFF" border="0" align="center" cellpadding="-2" cellspacing="-2" class="tableBorder">
          <tr align="center" valign="middle">
            <td height="24" colspan="2" background="Images/bg_login.gif"><font color="#505875">=== 用户登录 === </font> </td>
          </tr>
          <tr>
```

```
                <td width="61" height="27" align="right" valign="middle">用户名：</td>
                <td width="157" valign="middle"><input name="UID" type="text" maxlength="20"></td>
            </tr>
            <tr>
                <td height="27" align="right" valign="middle">密  码：</td>
                <td valign="middle"><input name="PWD" type="password"  maxlength="20" onKeyDown="if(event.keyCode==13)  mycheck(form_U,'
用户名')"></td>
            </tr>
            <tr align="center" valign="middle">
                <td height="27" colspan="2"><input name="Submit" type="button" class="btn_grey" value="登录" onClick="mycheck (form_U,'用户名')">
 
                <input name="Submit2" type="button" class="btn_grey" value="关闭" onClick="javascript:User.style.display='none';"></td>
              </tr>
          </table>
      </form>
</td>
</tr>
</table>
</div>
<!--应用层设计用户登录表单页面结束-->
```

（2）在页面中加入“用户登录”超链接，该超链接执行的操作是调用自定义 JavaScript 函数 Myopen()显示 User 层，关键代码如下：

```
<a href="#" onClick="Myopen(User)">用户登录</a>
```

（3）编写自定义的 JavaScript 函数 Myopen()，用于控制层的居中显示，关键代码如下：

```
<script language="javascript">
function Myopen(divID)
{
        //根据传递的参数确定显示的层
        divID.style.display='block';
        divID.style.left=(document.body.clientWidth-240)/2;
        divID.style.top=(document.body.clientHeight-139)/2;
}
</script>
```

秘笈心法

如果在网页中单独使用<div>而不加任何 CSS-P（Cascading Style Sheets Positioning 的简写，是 CSS 的一个扩展，它可以控制任何东西在网页上的位置），它在网页中的效果和使用<p>标记是一样的。

实例 451	应用 JavaScript 实现指定尺寸无边框无滚动条窗口 光盘位置：光盘\MR\16\451	高级 实用指数：★★★☆

实例说明

通常情况下，网页中弹出的窗口都是包括标题栏和有边框的，这样的窗口有时会影响到页面的整体效果。根据实际情况将弹出的窗口设置为无边框的窗口，会让页面更加引人注目，同时也可以统一网站风格。例如，明日实业网站的管理员登录窗口就是无边框窗口，运行本实例，单击页面下方的 TOP 超链接，即可弹出无边框的居中显示的管理员登录窗口，如图 16.35 所示，这样可以使页面看起来更加和谐统一。需要说明的是，该实例只适合 IE 6.0 及以下版本，对于已经安装更新版本（如更新版本为 SP1）的浏览器弹出的窗口不能指定尺寸，只能全屏显示。

关键技术

本实例主要应用 window 对象的 open()方法，弹出一个最大化的窗口，再通过 window 对象的 resizeTo()方法和 moveTo()方法控制窗口的尺寸和显示位置。下面对 window 对象的 resizeTo()方法进行详细介绍。

功能：将当前窗口改变成(x,y)大小。



图 16.35　应用 JavaScript 实现指定尺寸无边框无滚动条窗口

语法格式如下：

```
window.resizeTo(x,y)
```

参数说明

❶ x：表示水平宽度，单位为像素。

❷ y：表示垂直高度，单位为像素。

设计过程

（1）在实例的 index.htm 页面中，添加控制窗口弹出的超链接，本实例中采用的是图片热点超链接，该超链接执行的操作是调用自定义的 JavaScript 函数 manage()。manage()函数实现了在网页中弹出一个以最大化方式显示的窗口，该窗口的内容是 login.htm 页的内容，关键代码如下：

```
<img src="Images/re_top.gif" width="208" height="56" border="0" usemap="#Map">
<script language="javascript">
function manage()
{
	var w=window.open('login.htm','','fullscreen=1,scrollbars=0');
}
</script>
<map name="Map">
  <area shape="rect" coords="73,13,135,43" href="#" onClick="manage()">
</map>
```

（2）制作弹出的窗口页面 login.htm，在该页面中添加控制窗口位置和大小的 JavaScript 代码，关键代码如下：

```
<script language="javascript">
	self.resizeTo(321,203);
	width=screen.width;
	height=screen.height;
	self.moveTo((width-240)/2,(height-139)/2);
</script>
```

（3）在 login.htm 页面的<body>标记中还需要加入“scroll=no”，否则弹出的窗口会带有滚动条，关键代码如下：

```
<body scroll=no>
```

秘笈心法

window 对象的 resizeTo()方法在 IE 6.0 以上版本中，只能改变以一般状态显示的窗口的尺寸，不能改变最大化窗口（用 window 对象的 open()方法设置了 fullscreen=1 参数的窗口）的尺寸。

第17章

导航条的应用

- ⏭ 水平导航条的应用
- ⏭ 下拉菜单式导航条
- ⏭ 侧导航条设计

17.1 水平导航条的应用

随着网站内容的复杂化，应用导航条可以对网站内容根据类别进行分类，因此，导航条对于每个网站来说都是必不可少的。本节将介绍一些水平导航条的实例，水平导航条有很多种，如本节中介绍的带图标的文字导航条、Flash 导航条、图片按钮导航条等。

实例 452 带图标的文字导航条

光盘位置：光盘\MR\17\452

初级

实用指数：★★★

实例说明

在浏览一些网站时，有些网站的导航会采用文字和图标结合的方式，这样浏览者不通过文字也可以大概了解每个导航条的内容，增加网站的实用性，实例的运行效果如图 17.1 所示。

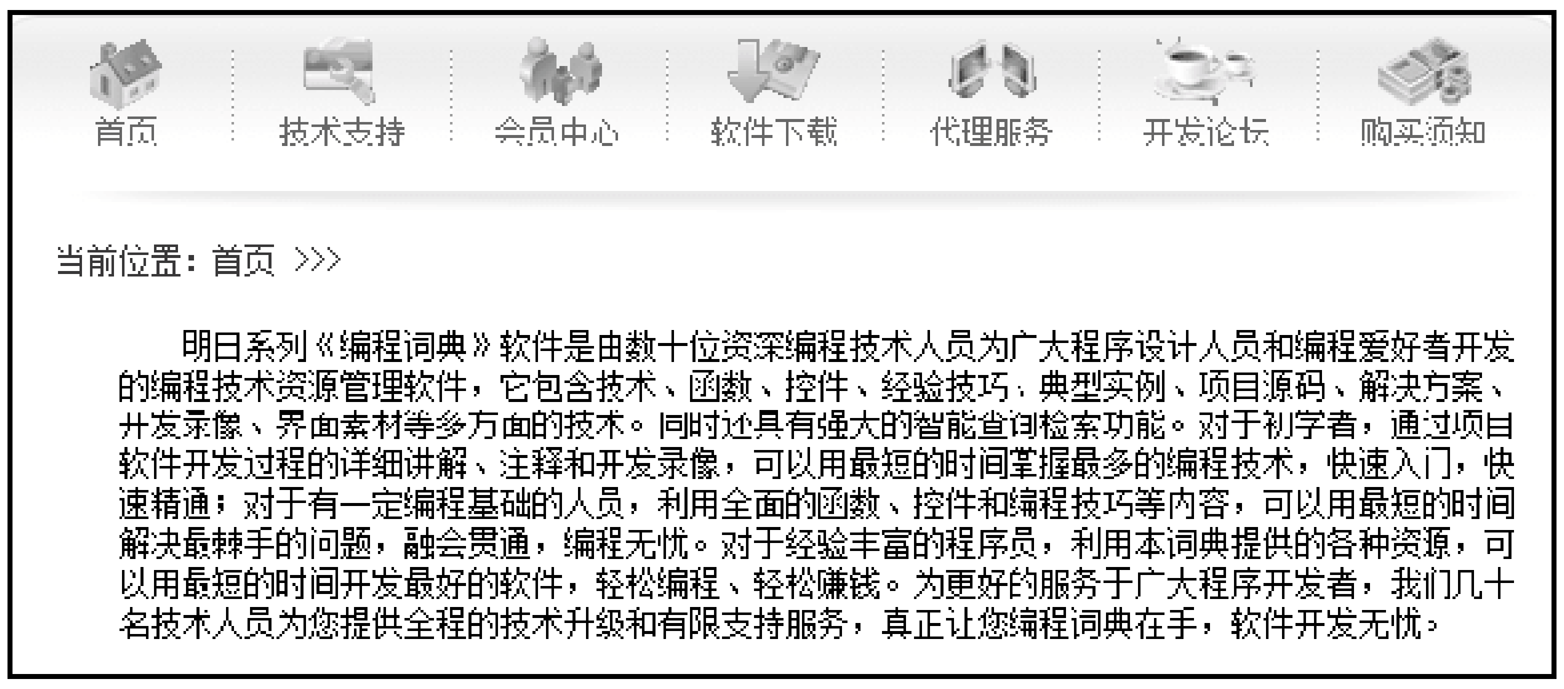


图 17.1 带图标的文字导航条

关键技术

本实例的实现比较简单，主要应用 HTML 的<img>标签和<a>标签来实现，应用<img>标签可以在网页中添加图片，<a>标签用于设置文字或其他内容的超链接。

<img>标签的常用属性如表 17.1 所示。

表 17.1 <img>标签的常用属性

属　性	说　明
src	设置图片的路径，路径可以是绝对路径，也可以是相对路径
alt	当图片无法显示时显示的字符串
id	<img>标签的唯一标识在 JavaScript 中常用 id 属性来操作 HTML 元素的对象
title	图片的提示信息，当鼠标停留在图片上时，显示 title 中设置的提示信息
border	设置图片的边框宽度
width	设置图片的宽度
height	设置图片的高度

<a>标签的常用属性如表 17.2 所示。

表 17.2 <a>标签的常用属性

属　性	说　明
href	用于指定目标文件的 URL 地址
name	指定超链接的名称

续表

属　性	说　明
title	该属性可选，用于指定指向超链接时显示的文字内容
target	该属性可选，用于指定超链接页面的打开方式，如果省略该属性，则目标文件将取代包含该超链接的文件。target 属性既可以是窗口或框架的名称，也可以是_blank、_parent、_self 和_top 4 个保留字中的一个

设计过程

（1）创建 index.jsp 页，在页面的适当位置添加一个一行多列的表格，该表格的列根据实际的导航链接的个数确定。

（2）在 index.jsp 页的表格的单元格中添加导航文字和图标，然后添加<a>超链接标签，把导航文字和图标作为<a>标签的内容，并设置<a>标签的链接地址，关键代码如下：

```
<td width="64"><a href="http://www.mingribook.com"><img src="images/T_mrbook.gif" width="42" height="28">明日图书</a></td>
```

秘笈心法

在超链接标签<a>中应用<img>标签时，显示的图片会带有蓝色边框，此时只需要将<img>标签的 border 属性设置为 0，即可去掉此边框。

实例 453　Flash 导航条

光盘位置：光盘\MR\17\453

初级　实用指数：★★★

实例说明

在一些个性网站中，网站导航的首选就是 Flash 导航条，Flash 导航条可以给浏览者带来更好的视觉效果，是网站个性的主要体现之一，本实例的运行效果如图 17.2 所示。



图 17.2　Flash 导航条

关键技术

本实例主要应用 Flash 动作脚本中 Button 类的 release()方法实现。release()方法在按下并释放鼠标左键时触发，语法格式如下：

```
on(release)
{
//此处插入语句
}
```

说明： 制作 Flash 导航条时需要应用有关 Flash 的软件制作工具，本实例应用的是 Macromedia Flash Professional 8。当然，读者也可以选择制作 Flash 的其他软件，如 Adobe Flash CS 系列的软件。

设计过程

（1）在 Macromedia Flash Professional 8 中，新建一个 Flash 文档，在菜单中选择“插入”→“新建元件”命令，在弹出的“创建新元件”对话框中选中“按钮”单选按钮，单击“确定”按钮。

（2）在“弹起”关键帧中输入文本“首页”，选择“指针经过”帧，单击鼠标右键，在弹出的快捷菜单中选择“插入关键帧”命令，在“按下”帧中插入一个动画元件，在“点击”帧中画出作用区域。

（3）返回场景，从库中将按元件拖入场景中适当的位置。

（4）选中按钮，打开“动作”面板，在代码区中输入如下代码：

```
on(release)
{
        getURL("index.jsp");
}
```

（5）依次做出多个按钮，然后生成 SWF 文件，并在网页中插入刚刚生成的 SWF 文件，代码如下：

```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
 codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=7,0,19,0" width="767" height="80">
        <param name="movie" value="navigation.swf" />
        <param name="quality" value="high" />
        <embed src="navigation.swf"  quality="high" pluginspage="http://www.macromedia.com/go/getflashplayer"  type="application/x-shockwave-flash"  width
="767" height="80"></embed>
      </object>
```

秘笈心法

在网页中插入 Flash 导航的 SWF 文件，应用的是<object>标签，其中 object 标签的 classid 属性值为支持 Flash 文件在网页中播放的固定值。

实例 454 图片按钮导航条

光盘位置：光盘\MR\17\454

初级

实用指数：★★★

实例说明

一些网站为了吸引更多的浏览者，导航条会设计成图片按钮的形式。本实例将介绍如何应用图片按钮作为网站的导航条，本实例的运行效果如图 17.3 所示。



图 17.3　图片按钮导航条

关键技术

本实例主要应用图片热点超链接来实现，应用 HTML 的<map>标签可以为图片添加热点。为图片设置热点超链接的语法格式如下：

```
<img src="img.gif" usemap="#MapName">
<map name="MapName">
  <area shape="value" coords="坐标" href="URL" alt="描述文字">
</map>
```

<map>标签的属性说明如表 17.3 所示。

表 17.3　<map>标签的常用属性

属　性	说　明
name	图片热点的名称，该属性值对应<img>标签的 usemap 属性值
alt	设定区域超链接的描述文字
shape	定义图片热点区域的形状，shape 属性包含 4 个属性值，分别为 rect、circle、poly 和 default
coords	设定区域坐标
href	设定区域的超链接地址

在<map>标签中，根据属性 shape 的取值不同，相应坐标的设定也不同。下面介绍 shape 属性的 3 种取值以及相应坐标的设定。

- ❑ 设定 shape 属性值为 rect

rect 属性值表示矩形区域，属性 coords 的坐标形式为（x1,y1,x2,y2）。其中，x1 和 y1 表示矩形左上角的 x 坐标和 y 坐标，x2 和 y2 表示矩形右下角的 x 坐标和 y 坐标。

- ❑ 设定 shape 属性值为 cicle

cicle 属性值表示圆形区域，属性 coords 的坐标形式为（x,y,r）。其中，x 和 y 为圆心坐标，r 为圆的半径。

- ❑ 设定 shape 属性值为 poly

poly 属性值表示多边形区域，属性 coords 的坐标形式为（x1,y1,x2,y2,…,xn,yn）。其中，xn 和 yn 代表构成多边形每一顶点的坐标值，多边形有几条边就有几对（x,y）坐标。

设计过程

（1）在 Dreamweaver 工具中，插入一个带有按钮的图片，然后在“属性”面板中选择“矩形热点工具”，为图片的每个按钮添加设定超链接。选中热点区域，在其“属性”面板的“超链接”文本框中输入超链接的地址，本实例中设置为空链接“#”，依次为每个热点设置超链接地址。

（2）新建 index.jsp 页，在该页中添加步骤（1）设置的图片热点超链接的代码，具体代码如下：

```
<body>
<table width="925" height="299" border="0" align="center" cellpadding="0" cellspacing="0">
  <tr>
    <td width="925" height="299"><img src="images/bg.jpg" width="869" height="336" border="0" usemap="#Map"></td>
  </tr>
</table>
<map name="Map">
  <area shape="rect" coords="292,17,382,57" href="#" alt="单击该按钮返回到商城首页">
  <area shape="rect" coords="410,19,500,59" href="#" alt="单击按钮进入到新品上市页面">
  <area shape="rect" coords="525,19,616,60" href="#" alt="单击按钮进入到特价商品页面">
  <area shape="rect" coords="640,18,734,59" href="#" alt="单击按钮进入到购物推车页面">
  <area shape="rect" coords="756,19,848,59" href="#" alt="单击按钮进入到订单查询页面">
</map>
</body>
```

秘笈心法

为了更准确地设置图片的热点区域坐标值，可以先利用制作网页的 Dreamweaver 工具，应用它的图形界面操作优势，可以准确地设置图片的热点坐标，然后再将生成的代码应用在 JSP 页中即可，这样省去了直接在 JSP 页中编写代码设置坐标值的麻烦。

实例 455	导航条的动画效果 光盘位置：光盘\MR\17\455	初级 实用指数：★★★

实例说明

导航条是网站设计中不可缺少的元素之一，它能正确地引导浏览者查找需要的资料，成为浏览者的网站路标。本实例实现的是导航条的动画效果，如图 17.4 所示，当鼠标经过导航文字时，导航文字将变成另一种效果。



图 17.4　导航条的动画效果

关键技术

本实例主要应用 JavaScript 控制<img>标签的 src 属性值来实现。当鼠标经过<img>标签表示的图片时，会

触发 onMouseMove 事件，该事件会调用相应的 JavaScript 代码来改变<img>标签的 src 属性值；当鼠标移出时，会触发 onMouseOut 事件，调用 JavaScript 代码修改<img>标签的 src 属性值为初始值。

在 JavaScript 中，可以应用 document 对象的 getElementById()方法获取<img>标签的对象，img1 为<img>标签的 id 值，代码如下：

```
document.getElementById("img1").src = "img1.gif";
```

设计过程

（1）创建 index.jsp 页，编写鼠标经过和移出的 JavaScript 方法 move()和 out()，关键代码如下：

```
function move(image,num){
        image.src='Images/top/menu_0'+num+'_over.gif';        //根据 num 参数值，更换图片
}
function out(image,num){
        image.src='Images/top/menu_0'+num+'.gif';             //根据 num 参数值，更换图片
}
```

（2）在 index.jsp 页中，应用<img>标签添加图片导航条，并设置<img>标签的 onMouseMove 事件调用 JavaScript 的 move()方法切换图片，onMouseOut 事件调用 out()方法还原图片，关键代码如下：

```
<td width="95" align="center">
<a href="#">
<img src="Images/top/menu_01.gif" id="image1"   border="0" onMouseMove="move(this,'1')" onMouseout="out(this,'1')">
</a>
</td>
<td width="95" align="center">
<a href="#">
<img src="Images/top/menu_02.gif" id="image2"   border="0" onMouseMove="move(this,'2')" onMouseout="out(this,'2')">
</a>
</td>
<td width="94" align="center">
<a href="#">
<img src="Images/top/menu_03.gif" name="image3"   border="0" onMouseMove="move(this,'3')" onMouseout="out(this,'3')">
</a>
</td>
```

秘笈心法

在<img>标签的 onMouseMove 事件和 onMouseOut 事件中调用 JavaScript 方法时，应用了 this 关键字作为参数，this 关键字表示一个对象，它表示<img>标签本身，应用它作为参数将图片对象传递到 JavaScript 方法中，可以方便地操作图片对象的属性。

实例 456	动态改变导航菜单的背景颜色 光盘位置：光盘\MR\17\456	初级 实用指数：★★★

实例说明

在浏览一些网站时，当鼠标经过导航菜单某一项时，其背景颜色将切换为其他颜色，实现这种简单的效果会更吸引浏览者的注意，本实例的运行效果如图 17.5 所示。



图 17.5　动态改变导航菜单的背景颜色

关键技术

本实例主要应用 JavaScript 方法来动态改变<td>标签的背景颜色。当鼠标经过<td>表示的导航菜单时，会触发 onMouseOver 事件，然后调用自定义的 JavaScript 方法改变<td>的背景颜色；当鼠标移出<td>时，会触发

onMouseOut 事件，调用自定义的 JavaScript 方法还原背景颜色为初始值。

在 JavaScript 中改变<td>标签的属性值时，需要为<td>设置一个 id 值，然后在 JavaScript 方法中，根据 document 对象的 getElementById()方法即可获取单元格对象，接下来就可以修改单元格对象的属性。如下代码展示了如何修改 id 为 td1 的单元格的背景颜色。

```
document.getElementById("td1").style.background="skyblue";
```

设计过程

（1）创建 index.jsp 页，编写鼠标经过事件的 JavaScript 方法 over()和鼠标移出事件的方法 out()，在这两个方法中，修改单元格的背景颜色，关键代码如下：

```
<script type="text/javascript">
function over(id){
	document.getElementById(id).style.background="skyblue";
}
function out(id){
	document.getElementById(id).style.background="white";
}
</script>
```

（2）在每个导航菜单的<td>标签中，设置 onMouseOver 事件调用 JavaScript 的 over()方法，设置 onMouseOut 事件调用 JavaScript 的 out()方法，关键代码如下：

```
<td width="64" id="td1" onmouseover="over('td1')" onmouseout="out('td1')"><a href="http://www.mingribook.com">明日图书</a></td>
<td width="64" id="td2" onmouseover="over('td2')" onmouseout="out('td2')"><a href="http://www.mingrisoft.com">明日软件</a></td>
<td width="64" id="td3" onmouseover="over('td3')" onmouseout="out('td3')"><a href="http://www.mingrisoft.com">关于明日</a></td>
<td width="64" id="td4" onmouseover="over('td4')" onmouseout="out('td4')"><a href="#">购买须知</a></td>
<td width="64" id="td5" onmouseover="over('td5')" onmouseout="out('td5')"><a href="http://www.mingribook.com">联系我们</a></td>
```

秘笈心法

在 JavaScript 方法中，修改单元格对象的背景颜色时，首先要调用 style 属性。需要注意的是，表示背景颜色的属性名为 background，而并不是 bgcolor。

实例 457　不用图片实现质感导航条　初级

光盘位置：光盘\MR\17\457　　实用指数：★★★

实例说明

一般质感比较强的导航条都是通过图像制作软件来制作质感图片的。本实例介绍的是只需要修改网页代码就可以改变导航条背景色的方法，即应用 CSS 样式实现，本实例的运行效果如图 17.6 所示。

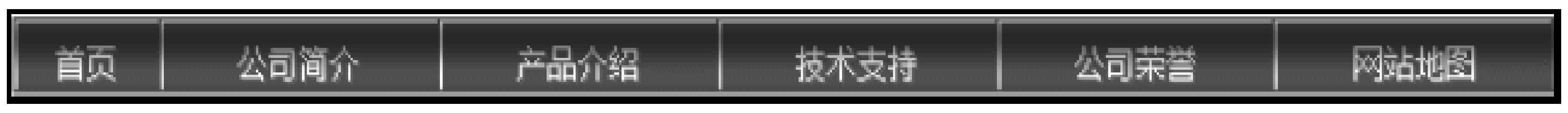


图 17.6　实现质感导航条

关键技术

本实例主要应用 CSS 样式的 wave 滤镜实现。wave 滤镜的语法格式如下：

```
{filter:wave(add=add,freq=freq,lightstrength=strength,phase=phase,strength=strength)}
```

参数说明

❶ add：表示是否要把对象按照波形样式打乱，取值为 true 或 false。

❷ freq：表示波形的频率，也就是指定在对象上共需要产生多少个完整的波纹。

❸ lightstrength：表示可以对波纹增强光影效果，取值范围在 0～100 之间。

❹ phase：用来设置正弦波的偏移量。

❺ strength：用来设置振幅的大小。

设计过程

（1）创建 index.jsp 页，在页面中导航位置添加一个单线边框的表格，并将表格的背景颜色设置为页面的主色调，同时添加导航文字，关键代码如下：

```
<table width="100%" height="27"   border="1" cellpadding="0" cellspacing="0"
  bordercolor="#FFFFFF" bordercolordark="#FFFFFF" bordercolorlight="#0066CC" bgcolor="#006666">
  <tr align="center">
    <td valign="bottom"><a href="#" class="shadow">首页</a></td>
    <td valign="bottom"><a href="#" class="shadow">公司简介</a></td>
    <td valign="bottom"><a href="#" class="shadow">产品介绍</a></td>
    <td valign="bottom"><a href="#" class="shadow">技术支持</a></td>
    <td valign="bottom"><a href="#" class="shadow">公司荣誉</a></td>
    <td valign="bottom"><a href="#" class="shadow">网站地图</a></td>
<td valign="bottom"><a href="#" class="shadow">与我联系</a></td>
</tr>
</table>
```

（2）为导航条设置 CSS 滤镜样式，也就是设置<table>标签的 style 属性，关键代码如下：

```
style="filter:wave(add=false,freq=1,lightstrength=50,phase=50)"
```

秘笈心法

可以将 CSS 样式代码写在<style>标签内，并设置样式的名称，然后为<table>标签的 class 属性引用该 CSS 样式名即可，这样有利于网站样式的维护，代码如下：

```
<style type="text/css">
.navigationStyle{
        filter:wave(add=false,freq=1,lightstrength=50,phase=50);
}
</style>
```

实例 458　标签页导航条

初级

光盘位置：光盘\MR\17\458

实用指数：★★★

实例说明

当网页内容比较复杂时，可以使用标签页导航条将功能类似的信息放在同一个标签页内，用户可以方便地在不同的标签页间进行切换。本实例将介绍如何实现标签页的导航条，本实例的运行效果如图 17.7 所示，单击“社会”、“法制”、“体育”和“娱乐”标签时，将分别显示不同的类别信息。



图 17.7　标签页导航条

关键技术

本实例主要应用 Dojo 工具包来实现，可以访问 Dojo 的官方网站 http://dojotookit.org 下载 Dojo 最新版本的工具包。Dojo 是一个 JavaScript 库，它的功能主要是处理 Ajax 请求，还包括对 DOM 的支持、对拖拉的支持等。Dojo 提供了一个美观而实用的工具集，包括树、富文本编辑器、日期选择器、菜单和容器等。本实例中应用的是 TabContainer 和 ContentPane 容器。

在使用 Dojo 时，首先需要加载所需包的对象。Dojo 对象的 require()函数可以调用某个包空间的对象。在使用 Dojo 时，如果需要的对象未引入 JavaScript 文件中，则应该调用 require()函数来请求这个包空间，Dojo 会根据 require 请求自动得到相应的 JavaScript 文件，并加载到内存中。Dojo 会自动维护已经加载的包列表，所以不会重复加载。require()函数的语法格式如下：

```
dojo.require(object)
```

参数说明

object：用于指定要调用的对象，包括包路径。

例如，加载 TabContainer 包的具体代码如下：

```
dojo.require("dijit.layout.TabContainer");
```

一个标签页效果需要由 TabContainer 和 ContentPane 两个容器组成，其中 TabContainer 表示整个标签页容器，而 ContentPane 则用于声明每一个标签页。在页面中声明 TabContainer 对象时，只要在<div>标签中设置 dojoType 属性为 dijit.layout.TabContainer 即可，代码如下：

```
<div id="mainTabContainer" dojoType="dijit.layout.TabContainer" style="width:541px;height:177px; display:block; border:0 solid #FFFFFF;">
```

然后还需要声明 ContentPane 对象，在声明 ContentPane 对象时，与声明 TabContainer 对象类似，需要指定 dojoType 属性值为 dijit.layout.ContentPane，通常还需要设置以下几个属性。

- refreshOnShow：用于指定在标签页从隐藏到展现时是否刷新数据，值为 true 或 false。
- href：用于指定该标签页内显示页面的 URL 地址。
- selected：用于指定该标签页内是否为选中状态，值为 true 或 false。
- title：用于指定单个标签页的标题。
- preload：用于指定是否强制加载数据，值为 true 或 false。

设计过程

（1）创建 index.jsp 页，在该页中导入 Dojo 提供的 CSS 样式表文件，以及 Dojo 库和 dijit 库，关键代码如下：

```
<style type="text/css">
    @import "js/dijit/themes/tundra/tundra.css";
    @import "js/dojo/resources/dojo.css";
</style>
<script type="text/javascript" src="js/dojo/dojo.js" djConfig="parseOnLoad: true"></script>
<script type="text/javascript" src="js/dijit/dijit.js"></script>
```

（2）在 JavaScript 代码中，应用 Dojo 对象的 require()函数导入 TabContainer 对象和 ContentPane 对象，关键代码如下：

```
<script type="text/javascript">
  dojo.require("dijit.layout.TabContainer");
  dojo.require("dijit.layout.ContentPane");
</script>
```

（3）应用<div>标签声明 TabContainer 和 ContentPane 对象，实现标签页导航，关键代码如下：

```
<div id="mainTabContainer" dojoType="dijit.layout.TabContainer" style="width:541px;height:177px; display:block; border:0 solid #FFFFFF;">
     <div  id="news1"  dojoType="dijit.layout.ContentPane"  refreshOnShow="true"  selected="true"   title="社会"  href="newsList.jsp?type=社会"
preload="true">社会新闻</div>
      <div id="news2" dojoType="dijit.layout.ContentPane" title="法制"   refreshOnShow="true" href="newsList.jsp?type=法制">法制新闻 </div>
      <div id="news3" dojoType="dijit.layout.ContentPane" title="体育" refreshOnShow="true" href="newsList.jsp?type=体育">体育新闻</div>
      <div  id="news4"  dojoType="dijit.layout.ContentPane"  title="娱乐"  refreshOnShow="true"  href="newsList.jsp?type=娱乐">  娱乐新闻
</div>
</div>
```

秘笈心法

Dojo 是一个非常庞大的 JavaScript 类库，Dojo 以包的形式组织众多功能，相同功能的函数、类、对象放在同一个包下，类似于 Java 中的类库。而导入包也类似于 Java 的 import，也可以使用（*）号，代码如下：

```
dojo.require("dojo.widget.*");
```

目前，在很多开发 Web 的项目中，都采用 Dojo 作为 JavaScript 框架，最主要是 Dojo 实现了 Ajax 的封装处理，应用它可以方便地处理 Ajax 请求。关于 Dojo 的详细介绍读者可以参考相关资料。

17.2　下拉菜单式导航条

下拉菜单式导航条可以使整个页面看起来更规范，本节将介绍一些常用的下拉菜单式导航条，如二级导航菜单、半透明背景的下拉菜单、弹出式下拉菜单等。

实例 459	二级导航菜单 光盘位置：光盘\MR\17\459	初级 实用指数：★★★

实例说明

网站中的导航条在整个网站中充当着网站路标的作用，一个层次分明的网站导航对于网站来说极为重要。由于网站内容过于复杂，有时一级导航菜单并不能明确划分网站内容，此时需要设计二级导航菜单。本实例实现的是如何为网站建立二级导航菜单，运行本实例，当鼠标移动到一级导航菜单时，将显示对应的二级导航菜单，如图 17.8 所示。



图 17.8　二级导航菜单

关键技术

实现二级导航菜单也很简单，应用 JavaScript 来动态处理 HTML 超链接<a>标签的鼠标事件即可实现。当鼠标经过一级导航菜单时，会触发 onMouseOver 事件，然后在该事件中调用自定义的 JavaScript 方法。在 JavaScript 方法中，应用 innerHTML 属性为<div>动态添加二级导航菜单信息即可。

例如，在网页中添加一个 id 为 mydiv 的<div>标签，应用 JavaScript 动态为该<div>添加内容的代码如下：

```
document.getElementById("mydiv").innerHTML="<a href="#">向 DIV 中添加内容</a>";
```

设计过程

（1）创建 index.jsp 页，编写动态添加二级导航菜单的 JavaScript 方法 addSecondMenu()，参数 value 用于判断一级导航类别的值，关键代码如下：

```
function addSecondMenu(value){
 var submenu = document.getElementById("submenu");
 switch (value){
	case "基础":
		submenu.innerHTML="<a href='#' target='mainF'>客户信息管理</a>|<a href='#' target='mainF'>商品信息管理</a>|<a href='#' target='mainF'>供应商信息管理</a>|<a href='#' target='mainF'>客户信息查询</a>|<a href='#' target='mainF'>商品信息查询</a>|<a href='#' target='mainF'>供应商信息查询</a>";
		break;
	case "采购":
		submenu.innerHTML="<a href='#' target='mainF'>商品采购</a>|<a href='#' target='mainF'>采购查询</a>";
		break;
	case "库存":
		submenu.innerHTML="<a href='#' target='mainF'>商品入库</a>|<a href='#' target='mainF'>商品入库退货</a>|<a href='#' target='mainF'>库存查询</a>|<a href='#' target='mainF'>价格调整</a>";
		break;
	…//此处省略了一些 case 语句的判断
 }
}
```

（2）在一级导航菜单的超链接<a>标签中，添加鼠标经过事件 onMouseOver，在该事件中调用添加二级导航菜单的方法 addSecondMenu()，关键代码如下：

```
<a href="#" onMouseOver="addSecondMenu('基础')">基础信息</a>|
<a href="#" onMouseOver="addSecondMenu('采购')">采购管理</a>|
<a href="#" onMouseOver="addSecondMenu('库存')">库存管理</a>|
<a href="#" onMouseOver="addSecondMenu('销售')">商品销售</a>|
<a href="#" onMouseOver="addSecondMenu('查询')">查询统计</a>|
<a href="#" onMouseOver="addSecondMenu('往来')">往来管理</a>|
<a href="#" onMouseOver="addSecondMenu('系统')">系统设置</a>
```

秘笈心法

本实例在 JavaScript 方法中应用的是 switch-case 语句，而不是 if 语句。当判断分支比较多时，应该考虑使用 switch-case 语句来实现。

实例 460　半透明背景的下拉菜单

光盘位置：光盘\MR\17\460

初级　实用指数：★★★

实例说明

本实例将介绍如何实现半透明背景的下拉菜单，运行本实例，当鼠标移动到一级导航菜单时，在其下方将显示出半透明的下拉式二级菜单，透过此下拉菜单仍可以看到页面上的内容，如图 17.9 所示。



图 17.9　半透明背景的下拉菜单

关键技术

实现半透明背景的下拉菜单，首先需要在页面中根据 onMouseOver 和 onMouseOut 事件调用 JavaScript 方法动态向页面中添加下拉菜单，然后再通过设置下拉菜单的 CSS 样式实现半透明效果。在 CSS 样式中，应用 alpha 滤镜来实现半透明的效果，alpha 滤镜的语法格式如下：

```
{filter:alpha(opacity=value,finishOpacity=value,style=value,startX=x1,startY=y1,finishX=x2,finishY=y2)}
```

参数说明

❶ opacity：表示透明度，取值范围在 0～100 之间，0 表示完全透明，100 表示完全不透明。

❷ finishOpacity：用于设置渐变的透明效果，可以使用该参数指定结束时的透明度，取值范围在 0～100 之间。

❸ style：指定透明区域的形状特征，取值分别为 0（统一形状）、1（线形）、2（放射状）、3（长方形）。

❹ startX：表示渐变透明效果区域的开始位置 x 坐标。

❺ startY：表示渐变透明效果区域的开始位置 y 坐标

❻ finishX：表示渐变透明效果区域的结束位置 x 坐标。

❼ finishY：表示渐变透明效果区域的结束位置 y 坐标。

设计过程

（1）创建 menu.js 文件，编写实现下拉菜单的 JavaScript 方法，主要包括控制下拉菜单的显示和隐藏的方法，关键代码如下：

```
var menuOffX=0;          //菜单距连接文字最左端距离
var menuOffY=18;         //菜单距连接文字顶端距离
var fo_shadows=new Array();
var linkset=new Array();
var IE=document.all&&navigator.userAgent.indexOf("Opera")==-1;
var netscape6=document.getElementById&&!document.all;
var netscape4=document.layers;
//初始化要显示的菜单
```

```
function showmenu(e,vmenu,mod){
        if (!document.all&&!document.getElementById&&!document.layers)
                return
        which=vmenu;
        clearhidemenu();
        IE_clearshadow();
        menuobj=IE? document.all.popmenu : netscape6? document.getElementById("popmenu") : netscape4?  document.popmenu : "";
        menuobj.thestyle=(IE||netscape6)? menuobj.style : menuobj;
        if (IE||netscape6)
                menuobj.innerHTML=which
        else{
                menuobj.document.write('<layer name="other" bgColor="#E6E6E6" width="165" onmouseover="clearhidemenu()"onmouseout= "hidemenu()" >'
+which+'</layer>');
                menuobj.document.close();
        }
        menuobj.contentwidth=(IE||netscape6)? menuobj.offsetWidth : menuobj.document.other.document.width
        menuobj.contentheight=(IE||netscape6)? menuobj.offsetHeight : menuobj.document.other.document.height
        eventX=IE? event.clientX : netscape6? e.clientX : e.x
        eventY=IE? event.clientY : netscape6? e.clientY : e.y
        var rightedge=IE? document.body.clientWidth-eventX : window.innerWidth-eventX;
        var bottomedge=IE? document.body.clientHeight-eventY : window.innerHeight-eventY;
        if (rightedge<menuobj.contentwidth)
                menuobj.thestyle.left=IE?  document.body.scrollLeft+eventX-menuobj.contentwidth+menuOffX  :  netscape6?window.pageXOffset+  eventX-
menuobj.contentwidth : eventX-menuobj.contentwidth
        else
                menuobj.thestyle.left=IE? IE_x(event.srcElement)+menuOffX : netscape6? window.pageXOffset+eventX : eventX
        if (bottomedge<menuobj.contentheight&&mod!=0)

 menuobj.thestyle.top=IE?document.body.scrollTop+eventY-menuobj.contentheight-event.offsetY+menuOffY-23                    :
netscape6?window.pageYOffset+eventY-menuobj.contentheight-10 : eventY-menuobj.contentheight
        else
                menuobj.thestyle.top=IE? IE_y(event.srcElement)+menuOffY : netscape6? window.pageYOffset+eventY+10 : eventY
        menuobj.thestyle.visibility="visible";
        IE_dropshadow(menuobj,"#999999",3);
        return false;
}
//计算 y 轴的坐标
function IE_y(e){
        var t=e.offsetTop;
        while(e=e.offsetParent){
                t+=e.offsetTop;
        }
        return t;
}
//计算 x 轴的坐标
function IE_x(e){
        var l=e.offsetLeft;
        while(e=e.offsetParent){
                l+=e.offsetLeft;
        }
        return l;
}
//显示菜单
function IE_dropshadow(el, color, size) {
        var i;
        for (i=size; i>0; i--){
                var rect = document.createElement('div');
                var rs = rect.style
                rs.position = 'absolute';
                rs.left = (el.style.posLeft + i) + 'px';
                rs.top = (el.style.posTop + i) + 'px';
                rs.width = el.offsetWidth + 'px';
                rs.height = el.offsetHeight + 'px';
                rs.zIndex = el.style.zIndex - i;
                rs.backgroundColor = color;
                var opacity = 1 - i / (i + 1);
                rs.filter = 'alpha(opacity=' + (100 * opacity) + ')';
                fo_shadows[fo_shadows.length] = rect;
        }
}
//清除显示
function IE_clearshadow(){
```

```
	for(var i=0;i<fo_shadows.length;i++){
		if (fo_shadows[i])
			fo_shadows[i].style.display="none"
	}
	fo_shadows=new Array();
}
//隐藏菜单
function hidemenu(){
	if (window.menuobj)
		menuobj.thestyle.visibility=(IE||netscape6)? "hidden" : "hide"
	IE_clearshadow()
}
//动态隐藏
function dynamichide(e){
	if (IE&&!menuobj.contains(e.toElement))
		hidemenu()
	else if (netscape6&&e.currentTarget!= e.relatedTarget&& !contains_netscape6(e.currentTarget, e.relatedTarget))
		hidemenu()
}
//延迟隐藏菜单
function delayhidemenu(){
	if (IE||netscape6||netscape4)
		delayhide=setTimeout("hidemenu()",500)
}
//停止延迟隐藏菜单
function clearhidemenu(){
	if (window.delayhide)
		clearTimeout(delayhide)
}
function highlightmenu(e,state){
	if (document.all)
		source_el=event.srcElement
	else if (document.getElementById)
		source_el=e.target
	if (source_el.className=="menuitems"){
		source_el.id=(state=="on")? "mouseoverstyle" : ""
	}
	else{
		while(source_el.id!="popmenu"){
			source_el=document.getElementById? source_el.parentNode : source_el.parentElement
			if (source_el.className=="menuitems"){
				source_el.id=(state=="on")? "mouseoverstyle" : ""
			}
		}
	}
}
//设置菜单背景
function overbg(tdbg){
	tdbg.style.background='url(Images/item_over.gif)'
	tdbg.style.border=' #9CA6C6 1px solid'
}
function outbg(tdbg){
	tdbg.style.background='url(Images/item_out.gif)'
	tdbg.style.border=''
}
var sysmenu='<table width=80><tr><td id=library onMouseOver=overbg(library) onMouseOut=outbg(library)><a href=#>图书馆信息</a></td></tr>\
<tr><td id=manager onMouseOver=overbg(manager) onMouseOut=outbg(manager)><a href=#>管理员设置</a></td></tr>\
<tr><td id=para onMouseOver=overbg(para) onMouseOut=outbg(para)><a href=#>参数设置</a></td></tr>\
<tr><td id=bookcase onMouseOver=overbg(bookcase) onMouseOut=outbg(bookcase)><a href=#>书架设置</a></td></tr>\
</table>'
var readermenu='<table width=90><tr><td id=readerType onMouseOver=overbg(readerType) onMouseOut=outbg(readerType)><a href=#>读者类型管理</a></td></tr>\
<tr><td id=reader onMouseOver=overbg(reader) onMouseOut=outbg(reader)><a href=#>读者档案管理</a></td></tr>\
</table>'
var bookmenu='<table width=90><tr><td id=bookType onMouseOver=overbg(bookType) onMouseOut=outbg(bookType)><a href=#>图书类型设置</a></td></tr>\
<tr><td id=book onMouseOver=overbg(book) onMouseOut=outbg(book)><a href=#>图书档案管理</a></td></tr>\
</table>'
var borrowmenu='<table width=60><tr><td id=Borrow onMouseOver=overbg(Borrow) onMouseOut=outbg(Borrow)><a href=#>图书借阅</a></td></tr>\
```

```
<tr><td id=renew onMouseOver=overbg(renew) onMouseOut=outbg(renew)><a href=#>图书续借</a></td></tr>\
<tr><td id=giveback onMouseOver=overbg(giveback) onMouseOut=outbg(giveback)><a href=#>图书归还</a></td></tr>\
</table>'
var querymenu='<table width=90><tr><td id=bookQuery onMouseOver=overbg(bookQuery) onMouseOut=outbg(bookQuery)><a href=#>图书档案查询</a></td></tr>\
<tr><td id=borrowQuery onMouseOver=overbg(borrowQuery) onMouseOut=outbg(borrowQuery)><a href=#>图书借阅查询</a></td></tr>\
<tr><td id=givebackQuery onMouseOver=overbg(givebackQuery) onMouseOut=outbg(givebackQuery)><a href=#>借阅到期提醒</a></td></tr>\
</table>'
var sortmenu='<table width=100><tr><td id=readerSort onMouseOver=overbg(readerSort) onMouseOut=outbg(readerSort)><a href=#>读者借阅排行榜</a></td></tr>\
<tr><td id=bookSort onMouseOver=overbg(bookSort) onMouseOut=outbg(bookSort)><a href=#>图书借阅排行榜</a></td></tr>\
</table>'
```

（2）创建 index.jsp 页，在页面的<script>标签中导入 menu.js 文件，然后在一级导航超链接<a>标签中设置 onMouseOver 和 onMouseOut 事件，当鼠标经过导航超链接时显示下拉菜单，鼠标移出时隐藏下拉菜单，关键代码如下：

```
<a   onmouseover=showmenu(event,sysmenu) onmouseout=delayhidemenu() class='navlink' style="CURSOR:hand" >系统设置</a> |
<a   onmouseover=showmenu(event,readermenu) onmouseout=delayhidemenu() class='navlink'style="CURSOR:hand" >读者管理</a> |
<a   onmouseover=showmenu(event,bookmenu) onmouseout=delayhidemenu() class='navlink' style="CURSOR:hand" >图书管理</a> |
<a   onmouseover=showmenu(event,borrowmenu) onmouseout=delayhidemenu() class='navlink' style="CURSOR:hand" >图书借还</a> |
<a   onmouseover=showmenu(event,querymenu) onmouseout=delayhidemenu() class='navlink' style="CURSOR:hand" >系统查询</a> |
<a   onmouseover=showmenu(event,sortmenu) onmouseout=delayhidemenu() class='navlink' style="CURSOR:hand" >排行榜</a> |
```

（3）定义名为 menuskin 的 CSS 样式，设置下拉菜单的半透明效果，关键代码如下：

```
.menuskin {
        BORDER: #666666 1px solid; VISIBILITY: hidden;
        POSITION: absolute;
        background-image:url("../Images/item_out.gif");
        background-repeat : repeat-y;
        Filter: Alpha(Opacity=85);
}
```

（4）在页面中添加一个<div>标签，动态加载的下拉菜单内容将显示在此标签体内，设置此标签应用半透明效果的样式，关键代码如下：

```
<div class=menuskin id=popmenu onmouseover="clearhidemenu();highlightmenu(event,'on')"
        onmouseout="highlightmenu(event,'off');dynamichide(event)"
        style="Z-index:100;position:absolute;"></div>
```

秘笈心法

alpha 属性是把一个目标元素与背景混合。设计者可以指定数据来控制混合的程度。这种“与背景混合”通俗地说是一个元素的透明度。通过指定坐标可以指定点、线、面的透明度。

实例 461 弹出式下拉菜单

光盘位置：光盘\MR\17\461

初级

实用指数：★★★

实例说明

所谓弹出式下拉菜单，是指类似于 Windows 应用程序的操作菜单，如 Word 的主菜单。本实例将介绍如何实现这种弹出式下拉菜单。运行本实例，当鼠标移动到一级导航菜单时，将弹出对应的二级下拉菜单，如果二级菜单中还有子菜单，鼠标经过时将继续显示子菜单，如图 17.10 所示。



图 17.10　弹出式下拉菜单

关键技术

本实例的弹出式菜单的效果主要是通过 Fireworks 实现的，具体方法将在设计过程中详细介绍，此处先对“弹出菜单编辑器”对话框中“内容”选项卡的“目标”栏中的可选值进行详细介绍。在“目标”栏中共有 4 个选项，各选项的功能如下。

- _blank：指定将链接的目标文件加载到未命名的新浏览器窗口中。
- _parent：指定将链接的目标文件加载到包含链接的父框架页或窗口中，如果包含链接的框架不是嵌套的，则链接的目标文件加载到整个浏览器窗口中。
- _self：指定将链接的目标文件加载到链接所在的同一框架或窗口中。
- _top：指定将链接的目标文件加载到整个浏览器窗口中，并由此删除所有框架。

设计过程

（1）在 Fireworks 中创建一个新的文件，名称为 popmenu.png，画布大小为 750×33。在该文件中添加 5 个图层，并在每一层中添加一个导航图片，如图 17.11 所示。选择“矩形热点工具”为每一张图片添加热点，完成后的效果如图 17.12 所示。



图 17.11　添加导航图片



图 17.12　添加热点

（2）在“读者管理”热点上单击鼠标右键，在弹出的快捷菜单中选择“添加弹出菜单”命令，将弹出“弹出菜单编辑器”对话框。在“文本”栏中输入条目内容，在“链接”栏中输入此条目所链接到的页面的文件名或地址，在“目标”栏中指定链接到的内容在何处显示，在该栏中有 4 个可选项，具体说明请读者参见本实例的关键技术部分。如果某一菜单项还包括子菜单，可以直接在相应位置输入子菜单项的名称，然后单击该窗口中的按钮即可。设置完成后的“读者管理”菜单内容如图 17.13 所示。

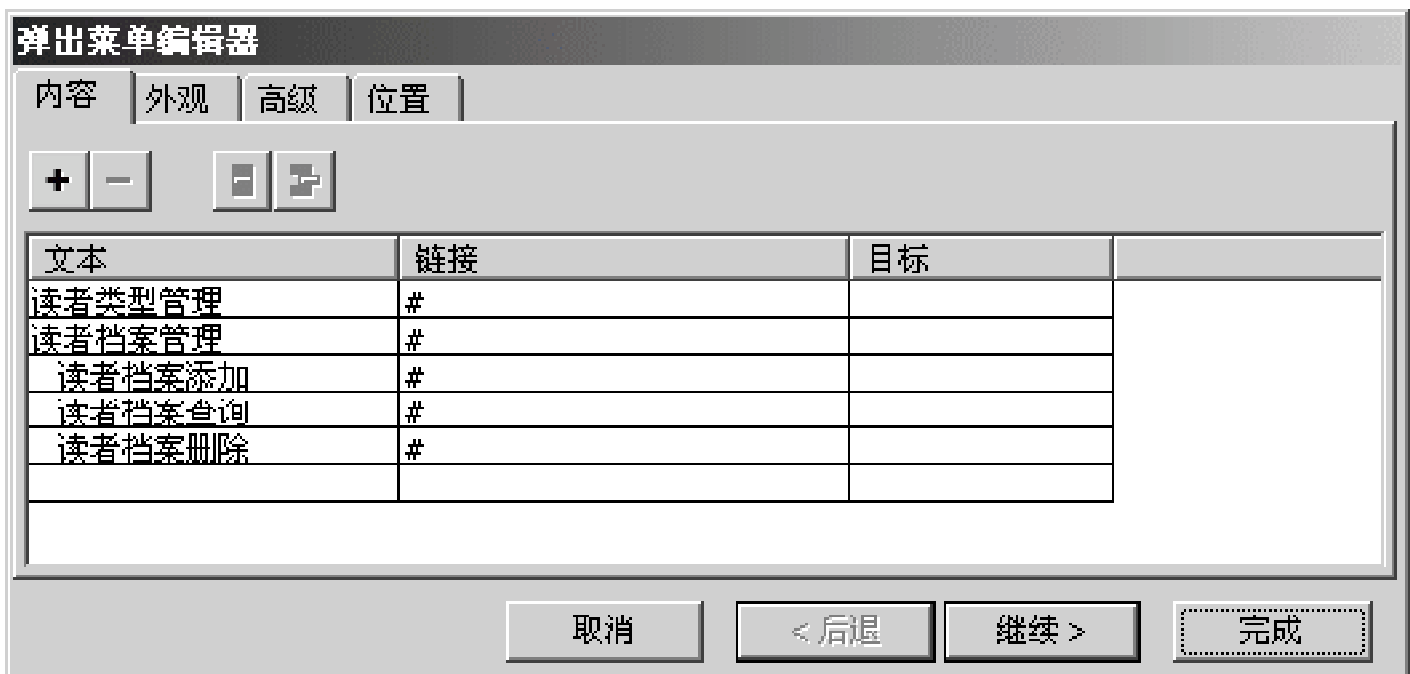


图 17.13　“弹出菜单编辑器”对话框

（3）单击“继续”按钮，设置菜单的外观，设置完成后再单击“继续”按钮，对菜单的边框和单元格进行详细设置，再单击“继续”按钮，设置菜单和子菜单的位置，最后单击“完成”按钮，完成一个菜单的设置。

（4）重复步骤（2）和步骤（3）的操作，给其他菜单添加下拉菜单。完成后按 F12 键进行预览。

（5）选择“文件”→“导出”命令将弹出“导出”对话框，在“文件名”文本框中输入文件名后，单击“保存”按钮即可。

（6）将设置完成后的菜单应用到网页中。此时读者需要将生成的 HTML 文件打开，将<body></body>之间的内容复制到指定网页中需要显示导航条的位置。如果用户想改变超链接的地址，可以直接在代码中修改，也就是将代码 location='#'中的“#”号替换为超链接地址。

秘笈心法

应用 Fireworks 图形界面工具可以在短时间内设计出弹出式菜单，并且会生成相应的 HTML 代码，只需要将这些 HTML 内容应用在网页中即可，这样省去了编写大量 HTML 代码的麻烦，从而提高开发效率。

实例 462 弹出式悬浮菜单

光盘位置：光盘\MR\17\462

初级

实用指数：★★★

实例说明

采用弹出式悬浮菜单，不但可以使网站的导航内容更加清晰，而且不影响页面的整体效果。运行本实例，当鼠标移动到一级导航菜单的标题上时，将弹出悬浮菜单显示该菜单对应的子菜单，鼠标移出时，将隐藏悬浮菜单，如图 17.14 所示。

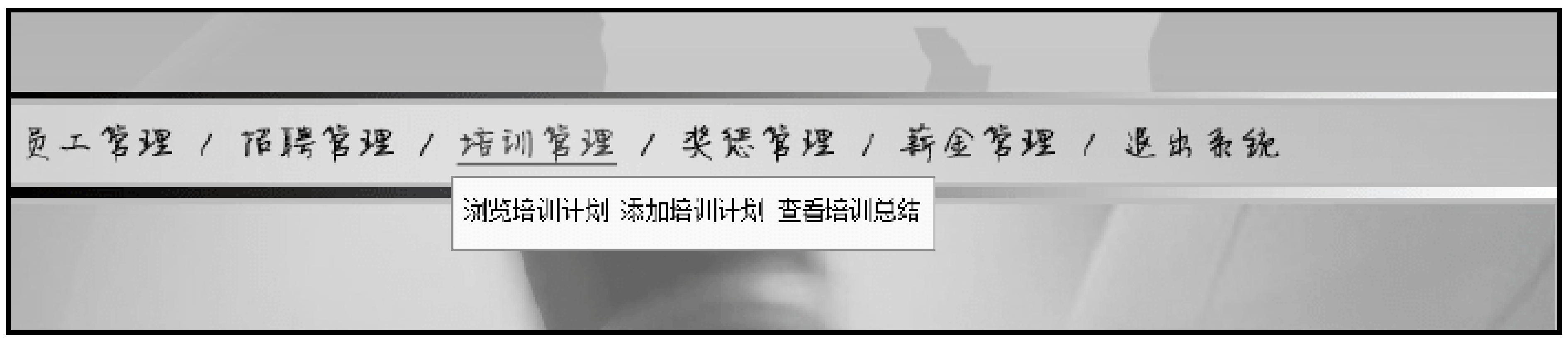


图 17.14　弹出式悬浮菜单

关键技术

本实例主要是在 JavaScript 中，动态改变<div>标签对象的 style 属性的 display 属性值来实现动态显示和隐藏二级导航菜单。其实，每一个一级菜单下的二级菜单内容已经添加在网页的<div>标签中，只是此时设置了<div>不显示。所以，在 JavaScript 中，当鼠标经过某个导航的标题时，只需要调用指定的<div>对象，动态修改它的 display 属性即可。display 属性包含两个用于显示和隐藏的属性值，分别为 none（隐藏）和 block（显示）。

例如，在网页中有一个 id 为 mydiv 的<div>标签，并设置了此<div>为隐藏。在 JavaScript 中，控制此<div>显示的语法格式如下：

```
document.getElementById("mydiv").style.display="block";
```

设计过程

（1）编写用于显示和隐藏的 JavaScript 方法，当鼠标经过一级菜单标题时会显示二级菜单，当鼠标移出时会隐藏二级菜单，关键代码如下：

```
function change(img,subMenu,path,type){
        img.src="images/"+path+".GIF";
        if(subMenu!=null){
                if(type==0){
                        subMenu.style.display="none";
                }else{
                        subMenu.style.display="block";
                }
        }
}
```

（2）在页面中，预先在<div>标签中为每个一级导航菜单添加二级菜单的内容，并设置二级菜单的<div>标签为隐藏，关键代码如下：

```
<div id="NUser" style="background-color:#F3FFD5; border:#75A102 1px solid; width:200px; position:absolute; display:none; left:1px; top: 34px;">
        <table width="100%" border="0" height="35px" cellspacing="0" cellpadding="0">
                <tr>
                        <td align="center"><a href="#">浏览员工信息</a>  <a href="#">添加新员工</a></td>
                </tr>
        </table>
</div>
…//此处省略了其他二级菜单的<div>内容
```

（3）在一级菜单的表格的<td>中设置 onMouseOver 和 onMouseOut 事件，调用步骤（1）中定义的 JavaScript 的 change()方法，动态改变二级菜单<div>的显示和隐藏，关键代码如下：

```
<td onMouseOver="change(ImgUser,NUser,'NUser_r',1)" onMouseOut="change(ImgUser,NUser,'NUser_b',0)" style="cursor:hand;">
       <img id="ImgUser" src="images/NUser_b.GIF" width="106" height="48" border="0">
```

```
        …//省略了二级菜单的<div>代码
</td>
```

秘笈心法

在本实例中，onMouseOver 和 onMouseOut 事件都是调用的同一个方法，只是由于传递的 type 参数不同，然后在 JavaScript 方法中，根据 type 参数值即可判断执行的是显示还是隐藏的代码。

实例 463　应用 setTimeout()函数实现展开式导航条　初级

光盘位置：光盘\MR\17\463

实用指数：★★★

实例说明

在有些网站中，会将其导航条设计为展开式导航条，这样可以更加引人注意。本实例将介绍如何应用 JavaScript 的 setTimeout()函数实现展开式导航条。运行本实例，页面右侧的“联系我们”导航条是慢慢展开的，并且每次弹出属性页面时，都会以同样的动画效果展开，如图 17.15 所示。



图 17.15　应用 setTimeout()函数实现展开式导航条

关键技术

本实例主要是应用 JavaScript 来动态改变<img>图片对象的高度，这样就需要在自定义 JavaScript 方法中，应用 setTimeout()函数间隔指定的时间动态修改图片对象的高度，从而实现动画效果。setTimeout()函数的语法格式如下：

```
window.setTimeout(function, time);
```

参数说明

❶ function：该参数为 setTimeout()函数所调用的自定义的 JavaScript 方法名或可执行的 JavaScript 代码。

❷ time：当过了 time 时间后调用指定的函数，单位为毫秒。

设计过程

（1）在网页中要显示“联系我们”导航条的位置，添加一个全部展开后的“联系我们”导航图片，并将图片的 height 属性值设置为 0，name 属性值设置为 our。在需要设置超链接的文字上添加图片并设置相应的超链接。

（2）编写 JavaScript 方法，应用 setTimeout()函数改变图片对象的高度值，实现图片的展开效果，关键代码如下：

```
function ourmove(){
        if(our.height<163){
                our.height=our.height+9
                setTimeout(ourmove,100)
        }
}
```

（3）在页面的<body>标签中，设置 onload 事件调用自定义的 JavaScript 方法 ourmove()，关键代码如下：

```
<body onload="ourmove()">
```

秘笈心法

由于 setTimeout()函数只能执行一次，因此在实现定时调用时，需要将 setTimeout()函数写在被调用函数的

函数体内，从而实现定时调用。

实例 464 应用 setInterval()函数实现展开式导航条

光盘位置：光盘\MR\17\464

初级

实用指数：★★★

实例说明

在有些网站中，会将其导航条设计为展开式导航条，这样可以更加引人注意。本实例将介绍如何应用 JavaScript 中的 setInterval()函数来实现展开式导航条。运行本实例，页面右侧的“联系我们”导航条是慢慢展开的，并且每次弹出属性页面时，都会以同样的动画效果展开，如图 17.16 所示。

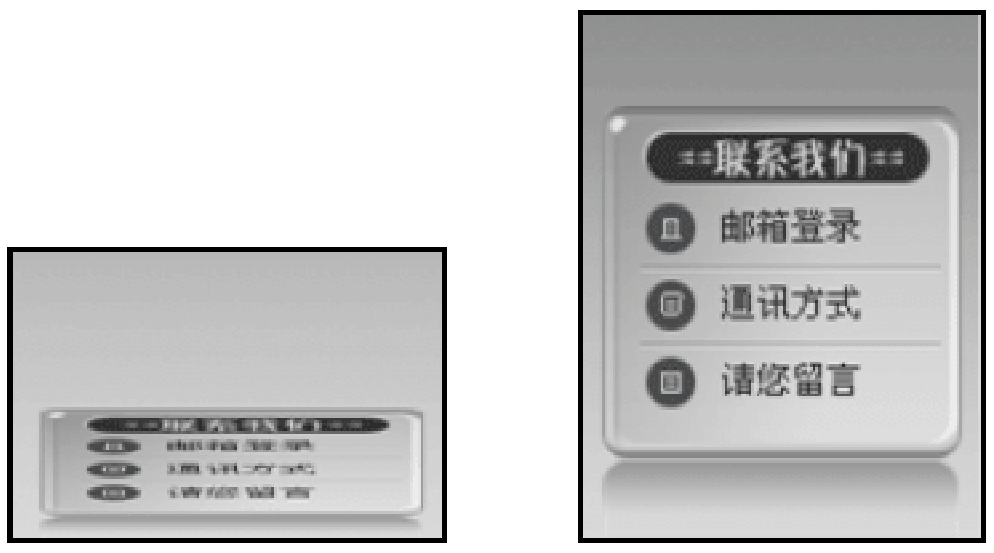


图 17.16　应用 setInterval()函数实现展开式导航条

关键技术

setInterval()函数相当于 JavaScript 中的定时器，使用它也可以定时执行 JavaScript 代码或函数，并一直执行。setInterval()函数的语法格式如下：

```
Object TimeID = window.setInterval(function, time);
```

参数说明

❶ TimeID：setInterval()函数的返回值，表示此定时器的唯一标识。当然，也可以不必设置返回值。

❷ function：表示所调用的自定义的 JavaScript 方法名或可执行的 JavaScript 代码。

❸ time：每间隔 time 时间调用一次指定的函数或 JavaScript 代码，单位为毫秒。

当需要取消定时器时，调用的是取消定时器的函数 clearInterval()，该函数需要根据设置的 setInterval()定时器返回值来取消定时器，代码如下：

```
TimeID = window.setInterval(test,1000);
window.clearInterval(TimerID);
```

设计过程

（1）在网页中要显示“联系我们”导航条的位置，添加一个全部展开后的“联系我们”导航图片，并将图片的 height 属性值设置为 0，name 属性值设置为 our。在需要设置超链接的文字上添加图片并设置相应的超链接。

（2）编写 JavaScript 方法，应用 setInterval()定时器函数定时改变图片对象的高度值，实现图片的展开效果，关键代码如下：

```
<script language="javascript">
  function ourmove(){
      if(our.height<163){
              our.height=our.height+9;
      }
  }
  window.setInterval(ourmove, 100);
</script>
```

秘笈心法

setTimeout()函数与 setInterval()函数的区别在于：setTimeout()函数只执行一次，如果希望继续执行，只能写在被调用函数的函数体内；setInterval()函数会根据指定间隔时间一直执行下去，并且是自动执行的，不需要手

动调用。

实例 465　用层制作下拉菜单 1

光盘位置：光盘\MR\17\465

初级

实用指数：★★★

实例说明

本实例通过层实现下拉菜单，当鼠标经过导航标题时，会在下面以下拉方式显示下拉列表，并可以通过选择下拉列表中的选项进入指定的网页，本实例的运行效果如图 17.17 所示。



图 17.17　用层制作的下拉菜单

关键技术

本实例主要应用 CSS 样式中的 display 属性来控制层是否显示，当层显示时，用 CSS 样式中的 rect()方法对层以 y 轴逐渐增大的方式进行局部显示。rect()方法用于检索或设置对象的可视区域，区域外的部分是透明的，必须将 position 的值设为 absolute，此属性方可使用。rect()方法的语法格式如下：

```
clip:auto|rect(number numner number number)
```

参数说明

❶ auto：对象无剪切。

❷ rect(number number number number)：依据（上→右→下→左）的顺序提供自对象左上角为（0,0）坐标计算的 4 个偏移数值，其中任一数值都可用 auto 替换，即此边不剪切。

设计过程

（1）在页面中编写用于改变下拉菜单颜色及位置的 CSS 样式，关键代码如下：

```
<style type="text/css">
.menubar{
        position:absolute;
        top:10px;
        width:100px;
        height:20px;
        cursor:default;
        border-width:1px;
        border-style:outset;
        color:#99FFFF;
        background:#669900
}
.menu{
        position:absolute;
        top:32px;
        width:140px;
        display:none;
        border-width:2px;
        border-style:outset;
        border-color:white sliver sliver white;
        background:#333399;
        padding:15px
}
.menu A{text-decoration:none;color:#99FFFF;}
.menu A:hover{color: #FFFFFF;}
</style>
```

（2）在页面的表格中，添加用于显示一级导航条的<div>标签，关键代码如下：

```
<tr>
<td width="20%">
        <div align="center"id="Tdiv_1" class="menubar" onmouseover="divControl(1)" onmouseout="divControl(0)">教育网站</div>
</td>
…       //由于代码类似，此处省略了其他导航<div>标签的代码
</tr>
```

（3）在页面中添加为一级导航设置的二级导航菜单选项，关键代码如下：

```
<tr>
<td width="20%">
        <div align="left"id="Div1" class="menu" onmouseover="keepstyle(this)" onmouseout="hidediv(this)">
                <a href="#">重庆 XX 大学</a><br>
                <a href="#">长春 XX 大学</a><br>
                <a href="#">吉林 XX 大学</a>
        </div>
</td>
…//此处省略了其他一级导航下对应的二级导航菜单内容
</tr>
```

（4）编写实现下拉菜单的 JavaScript 方法 divControl()，判断是否显示相对应的下拉列表，关键代码如下：

```
function divControl(show){        //判断是否显示相对应的下拉列表
        window.event.cancelBubble=true;
        var objID=event.srcElement.id;
        var index=objID.indexOf("_");
        var mainID=objID.substring(0,index);
        var numID=objID.substring(index+1);
        if(mainID=="Tdiv"){
                if(show==1)
                        eval("showdiv("+"Div"+numID+")");
                else
                        eval("hidediv("+"Div"+numID+")");
        }
}
```

编写自定义函数 displayMenu()，用于在显示下拉菜单时，以下拉方式显示下拉菜单，关键代码如下：

```
var nbottom=0,speed=2;
function displayMenu(obj){        //在显示下拉菜单时，以下拉方式显示下拉菜单
        obj.style.clip="rect(0 100% "+nbottom+"% 0)";
        nbottom+=speed;
        if(nbottom<=100)
                timerID=setTimeout("displayMenu("+obj.id+"),1");
        else
                 clearTimeout(timerID);
}
```

编写自定义函数 showdiv()，用于显示下拉列表的下拉选项，关键代码如下：

```
function showdiv(obj){      //显示下拉列表
        obj.style.display="block";
        obj.style.clip="rect(0 0 0 0)";
        nbottom=5;
        displayMenu(obj);
}
```

编写自定义函数 hidediv()，用于隐藏下拉列表的下拉选项，关键代码如下：

```
function hidediv(obj){      //隐藏下拉列表
        nbottom=0;
        obj.style.display="none";
}
```

秘笈心法

<div>标签经常用于动态加载网页内容，而且经常使用 JavaScript 控制<div>标签来实现各种特殊效果，如日期选择器以及 Dojo 工具包中的视图工具集等。

实例 466	用层制作下拉菜单 2 光盘位置：光盘\MR\17\466	初级 实用指数：★★★

实例说明

本实例实现的是用层制作的动态效果的下拉菜单。运行本实例，首先页面会显示“主菜单”，当鼠标移动到“主菜单”文字时，下面的菜单项会逐渐从两端移动到中间，并且由透明状态逐渐变为不透明，运行结果如

图 17.18 所示。

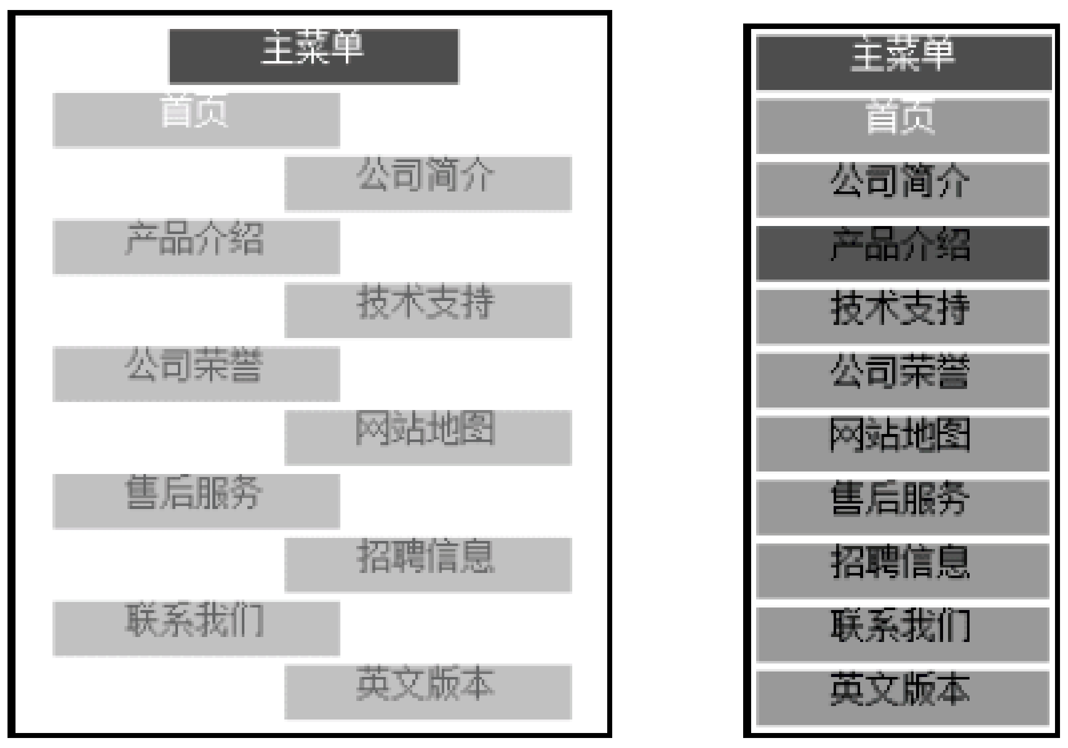


图 17.18　主菜单中的菜单项由两端逐渐移动到中间

关键技术

本实例主要是应用 CSS 样式的 opacity 属性控制层的透明度，用 pixelLeft 属性来控制层的横向位置。下面对 opacity 属性和 pixelLeft 属性进行详细说明。

- opacity：表示对象的透明度。取值范围在 0～100 之间，0 表示完全透明，100 表示完全不透明。
- pixelLeft：用于改变层的水平位置。只有在 position 属性设置为 absolute 时，该属性才有效。

设计过程

（1）创建 index.htm 页，在页面中预先添加表示导航菜单的层，用来显示下拉菜单中的各个选项，关键代码如下：

```
<div align=center id="menu01" class="menuStyle" onmouseover='stopTim();moveIn()' onmouseout='runTim()'>
<font color=white>主菜单</font></div>
<div align=center id="div1" class="divStyle" style="left:50px; top:55px;z-index:2; padding-top:4;"
        onmouseover="this.style.backgroundColor= '#006699';stopTim()"
        onmouseout="this.style.backgroundColor= '#00CC66';runTim()">首页</div>
<div align=center id="div2" class="divStyle" style="left:250px; top:77px; z-index:3; padding-top:3;"
        onmouseover="this.style.backgroundColor= '#006699';stopTim()"
        onmouseout="this.style.backgroundColor= '#00CC66';runTim()">公司简介</div>
<div align=center id="div3" class="divStyle" style="left:50px; top:99px;    z-index:4; padding-top:3;"
        onmouseover="this.style.backgroundColor= '#006699';stopTim()"
        onmouseout="this.style.backgroundColor= '#00CC66';runTim()">产品介绍</div>
<div align=center id="div4" class="divStyle" style="left:250px; top:121px;    z-index:5;padding-top:3;"
        onmouseover="this.style.backgroundColor= '#006699';stopTim()"
        onmouseout="this.style.backgroundColor= '#00CC66';runTim()">技术支持</div>
…//此处省略了其他菜单项的<div>标签代码
```

（2）编写用于实现下拉菜单的 JavaScript 代码。编写自定义函数 runTim()，用于在鼠标指向主菜单时显示菜单项，关键代码如下：

```
var Timer,timIn,timOut;
var leftLine = 50;
function runTim(){
        window.Timer = setTimeout('moveOut()',300);
}
```

编写自定义函数 stopTim()，用于在鼠标移出主菜单时隐藏菜单项，关键代码如下：

```
function stopTim(){
        clearTimeout(window.Timer);
}
```

编写自定义函数 moveIn()，将菜单项以奇数的形式从左右两边向中间移动，并使各菜单项的透明度不断加大，关键代码如下：

```
function moveIn(){
      if( leftLine != 150)        {
        div1.style.pixelLeft += 20; div1.filters.alpha.opacity += 20;
        div2.style.pixelLeft -= 20; div2.filters.alpha.opacity += 20;
        div3.style.pixelLeft += 20; div3.filters.alpha.opacity += 20;
```

```
        div4.style.pixelLeft -= 20; div4.filters.alpha.opacity += 20;
        div5.style.pixelLeft += 20; div5.filters.alpha.opacity += 20;
        div6.style.pixelLeft -= 20; div6.filters.alpha.opacity += 20;
        div7.style.pixelLeft += 20; div7.filters.alpha.opacity += 20;
        div8.style.pixelLeft -= 20; div8.filters.alpha.opacity += 20;
        div9.style.pixelLeft += 20; div9.filters.alpha.opacity += 20;
        div10.style.pixelLeft -= 20; div10.filters.alpha.opacity += 20;
        leftLine += 20;
        }
        else{
                clearTimeout(window.timIn);
                return false;
        }
        timerIn=window.setTimeout('moveIn()',1);
}
```

编写自定义函数 moveOut()，将菜单项以奇数的形式从中间向左右两边移动，并使各菜单项的透明度不断减小，关键代码如下：

```
function moveOut(){
    clearTimeout(window.timIn);
    if (leftLine != 50){
        div1.style.pixelLeft -= 20; div1.filters.alpha.opacity -= 20;
        div2.style.pixelLeft += 20; div2.filters.alpha.opacity -= 20;
        div3.style.pixelLeft -= 20; div3.filters.alpha.opacity -= 20;
        div4.style.pixelLeft += 20; div4.filters.alpha.opacity -= 20;
        div5.style.pixelLeft -= 20; div5.filters.alpha.opacity -= 20;
        div6.style.pixelLeft += 20; div6.filters.alpha.opacity -= 20;
        div7.style.pixelLeft -= 20; div7.filters.alpha.opacity -= 20;
        div8.style.pixelLeft += 20; div8.filters.alpha.opacity -= 20;
        div9.style.pixelLeft -= 20; div9.filters.alpha.opacity -= 20;
        div10.style.pixelLeft += 20; div10.filters.alpha.opacity -= 20;
        leftLine -= 20;
        }
        else{
                clearTimeout(window.timOut);
                return false;
        }
        timerOut=window.setTimeout("moveOut()",1);
}
```

秘笈心法

本实例实现菜单项从两侧逐渐向中间移动时，主要是应用了 JavaScript 中的 setTimeout()函数，间隔一段时间就修改一次 opacity 和 pixelLeft 属性值，从而实现预期效果。

17.3 侧导航条设计

在网站设计时，有时为了体现页面的整体效果，可以将网站导航条放置在侧面。本节将介绍几个不同样式的侧导航条的实例。

实例 467	收缩式导航菜单 光盘位置：光盘\MR\17\467	初级 实用指数：★★★

实例说明

在网站中不仅可以设置导航条，而且还可以设置导航菜单。由于菜单内容比较多，同一页面显示比较杂乱，所以很多设计者都采用收缩式的菜单形式。运行本实例，单击“网站管理”超链接时，页面将展开“网站管理”

的子菜单，再次单击“网站管理”超链接时，展开的导航菜单将收缩为初始状态，如图 17.19 所示。

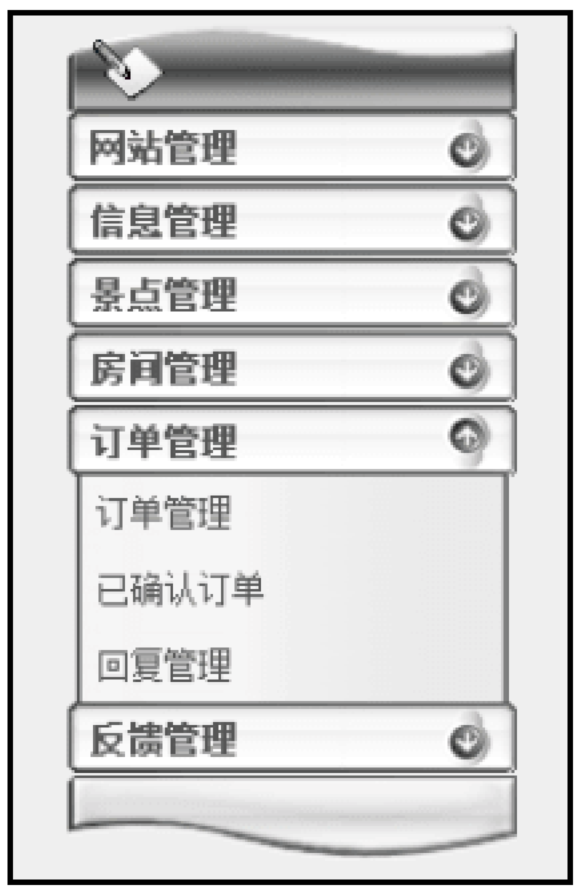


图 17.19　收缩式导航菜单

关键技术

本实例主要是应用 JavaScript 控制显示和隐藏表格的功能，实现收缩式导航菜单的功能。单击导航超链接，显示当前菜单的内容，隐藏上一个显示的菜单，在隐藏菜单时，让其有规律地隐藏，从而实现展开或收缩的动画效果。

设计过程

（1）创建 index.jsp 页，在页面中添加一级导航菜单项以及二级导航菜单，关键代码如下：

```
<tr style="CURSOR: hand">
            <td class="list_title" id="list1" onmouseover="this.typename='list_title2';"   onclick="change(menu1,50,list1);"
               onmouseout="this.typename='list_title';"   background="images/title_show.gif" height="25">
                 <span>网站管理</span>
        </td>
</tr>
<tr>
     <td align="center" valign="middle">
        <div class=sec_menu id=menu1 style="DISPLAY: none; width: 158px; height: 0px">
                 <table   cellSpacing="0" cellPadding="0" width="152" align="center"    background="images/bg.gif" style="padding-left:5px">
                    <tr><td height="25"><a href="#" target="BoardList">更改初始信息</a></td></tr>
                    <tr><td height="25"><a href="#" target="BoardList">查看服务器信息</a></td></tr>
                 </table>
        </div></td>
</tr>
…//此处省略了其他一级菜单以及二级菜单的内容
```

（2）编写展开菜单项的自定义 JavaScript 方法 show()，关键代码如下：

```
function show(obj,maxg,obj2){
        if(obj.style.pixelHeight<maxg){
                obj.style.pixelHeight+=maxg/10;
                obj2.background="images/title_hide.gif";              //改变菜单标题的背景
                if(obj.style.pixelHeight==maxg/10){
                        obj.style.display='block';                    //设置指定菜单项显示
                }
                myObj=obj;
                mymaxg=maxg;
                myObj2=obj2;
                setTimeout('show(myObj,mymaxg,myObj2)','5');          //每隔一段时间调用一次 show()函数，用于实现渐渐展开效果
        }
}
```

（3）编写收缩菜单项的自定义方法 hide()，关键代码如下：

```
function hide(obj,maxg,obj2){
        if(obj.style.pixelHeight>0){
                if(obj.style.pixelHeight==maxg/5){
                        obj.style.display='none';                     //设置指定菜单项隐藏
                }
```

```
            obj.style.pixelHeight-=maxg/5;
            obj2.background="images/title_show.gif";          //改变菜单标题的背景
            myObj=obj;
            mymaxg=maxg
            myObj2=obj2;
            setTimeout('hide(myObj,mymaxg,myObj2)','5');     //每隔一段时间调用一次 hide()函数，用于实现渐渐收缩效果
      }else if(whichContinue){
             whichContinue.click();
      }
}
```

（4）编写自定义方法 change()，实现当单击菜单标题时，隐藏前一个展开的菜单项，显示当前菜单项，关键代码如下：

```
function change(obj,maxg,obj2){
      if(obj.style.pixelHeight){
            hide(obj,maxg,obj2);                              //收缩菜单项
            nopen=";
            whichcontinue=";
      }else if(nopen){                                        //收缩已经展开的菜单项
             whichContinue=obj2;
             nopen.click();
      }else{
             show(obj,maxg,obj2);                             //展开菜单项
             nopen=obj2;
            whichContinue=";
      }
}
```

秘笈心法

本实例主要是在 JavaScript 中，修改二级菜单<div>标签 style 样式的 pixelHeight 属性值来实现菜单的收缩与展开效果。

实例 468 | 树状导航菜单 | 高级

光盘位置：光盘\MR\17\468 | 实用指数：★★★

实例说明

对于一个导航文字很多，并且可以对导航内容进行分类的网站来说，可以将页面中的导航文字以树状图的形式显示。树状图的导航菜单在实际开发应用中非常多，应用它可以方便用户查看。运行本实例，单击节点名称前的加号“+”可以展开指定的节点，单击减号“-”可以收缩子节点，如图 17.20 所示。



图 17.20　树状导航菜单

关键技术

本实例主要是通过 JavaScript 控制表格行的<tr>标签的显示或隐藏来实现节点的显示和隐藏。控制<tr>标签的显示和隐藏，主要是在 JavaScript 中控制<tr>标签对象的 display 属性来实现的。display 属性前面已经介绍过，所以此处不再赘述。

设计过程

（1）利用 JavaScript 定义一个函数 ShowTR()，用于显示或隐藏表格中指定行的内容，关键代码如下：

```
function ShowTR(objImg,objTr){
      if(objTr.style.display == ""){                         //折叠展开的节点
            objTr.style.display = "none";
            objImg.src = "images/jia.gif";                    //改变图标
            objImg.alt = "展开";                              //设置工具提示为“展开”
```

```
		}else{					//展开折叠的节点
			objTr.style.display = "";
			objImg.src = "images/jian.gif";		//改变图标
			objImg.alt = "折叠";				//设置工具提示为“折叠”
		}
}
```

（2）为了实现在默认情况下第一个节点为展开状态，可以在该函数后面加入以下代码，用于当第一个节点包括子节点时，设置第一个节点为展开状态，关键代码如下：

```
<%
	if(m>1)	//设置第一个节点为展开状态
		out.println("<script language='javascript'>ShowTR(img1,OpenRep1)	//设置第一个节点为展开状态</script>");
%>
```

（3）从视图 V_Type 中查询商品的分类信息，关键代码如下：

```
<%
	ConnDB conn=new ConnDB();
	ResultSet rs_superType=conn.executeQuery("select ID,superType from V_Type group by ID,superType order by superType desc");
%>
```

（4）循环显示商品分类信息，并查询各分类中所包括的子类名称，如果该分类有相对应的子类，则在节点前显示加号“+”，否则显示减号“-”。注意：需要为每个包含子节点的节点设置超链接，该超链接执行操作时调用自定义的 JavaScript 函数 showTR()，实现节点的展开与折叠。关键代码如下：

```
<table width="100%" height="40"	border="0" cellpadding="0" cellspacing="0">
	<%
		int ID_superType=0;
		String superType="";
		int m=1;
		while(rs_superType.next()){
			ID_superType=rs_superType.getInt(1);
			superType=rs_superType.getString(2);
			String sql="select * from V_Type where ID="+ID_superType;
			ResultSet rs_subType=conn.executeQuery(sql);
			String subType="";
			int subID=0;
			//获取记录总数
			rs_subType.last();
			int subType_RC=rs_subType.getRow();
			rs_subType.first();
		 %>
		 <tr>
		<td>
			<%if(subType_RC<=0){%>
			<img src="images/jian_null.gif" width="38" height="16" border="0">
			<%=superType%>
			<%}else{%>
			<a href="Javascript:ShowTR(img<%=m%>,OpenRep<%=m%>)">
				<img src="images/jia.gif" border="0" alt="展开" id="img<%=m%>">
			</a>
			<a href="Javascript:ShowTR(img<%=m%>,OpenRep<%=m%>)"><%=superType%></a>
			<%}%>
		</td>
		<%if (subType_RC>0){%>
		  <tr id="OpenRep<%=m%>" style="display:none;">
		   <td colspan="6">
			<%do{
				subType=rs_subType.getString(3);
				subID=rs_subType.getInt(4);
			%>
			<table width="100%"	border="0" cellspacing="-2" cellpadding="-2">
			  <tr>
			      <td height="25" align="center">
				<table width="90%"	border="0" cellspacing="0" cellpadding="0">
				<tr onMouseOver="this.style.background='#EEEEEE'"
					onMouseOut="this.style.background=''">
				<td width="15%"> </td>
				<td width="10%" align="center">
					<img src="images/folder.gif" width="16" height="16" border="0">
```

```
                    </td>
                    <td width="75%"><a href="#"><%=subType%></a></td>
                    </tr>
                    </table>
                    </td>
                </tr>
              </table>
                <%   m++;   //注意，该条语句一定不能少
                    }while(rs_subType.next());
              %>
          </td>
          <%}%>
        </tr>
      <%}%>
</table>
```

秘笈心法

HTML 元素中的<table>、<tr>、<td>、<img>、<form>、<imput>和<button>等标签都具有 display 属性，读者可以根据实际情况进行设置。

实例 469 自动隐藏的弹出式菜单

光盘位置：光盘\MR\17\469

高级

实用指数：★★★

实例说明

自动隐藏式菜单简洁易用，在不使用时能自动隐藏，使页面更具有灵活性，为网页增添了不少亮丽的光彩。运行本实例，在网页的左端显示一个自动隐藏的弹出式菜单，当鼠标移动到菜单标题上时，将自动弹出导航菜单，单击其中的菜单项，将进入相应的网页链接地址，当鼠标移出菜单时，导航菜单将自动隐藏，如图 17.21 所示。



图 17.21　自动隐藏的弹出式菜单

关键技术

本实例主要利用 JavaScript 来控制<div>层的显示和隐藏。当鼠标经过<div>层时会触发 onMouseOver 事件，然后在 JavaScript 中设置<div>的 display 属性值为 block（显示层），当鼠标移出 <div>层时触发 onMouseOut 事件，设置 display 属性值为 none（隐藏层）即可。

设计过程

（1）创建 index.jsp 页，在页面中编写用于显示导航菜单的<div>标签，在该标签中添加一个表格，然后用 JavaScript 代码向表格中添加每个菜单项，关键代码如下：

```
<div style="position:absolute; float:left; z-index:100;top: 149px;"onMouseOver="mouseover('menu')" onMouseOut="mouseout('menu')">
    <div id="menu" style="float:left; text-align:center; width:70px; height:302px; padding-top:5px; display:none;background-image:url(images/menu_
bg.gif)">
      <table border="0" cellpadding="0" cellspacing="1" bgcolor="#666666" style="padding:5px;">
          <script language="JavaScript">
            var sitems=["图书介绍","新书预告","图书销售","勘误发布","资料下载","好书推荐","技术支持","联系我们"];//菜单名称

            var sitemlinks=new Array();              //链接地址
            for(var i=0;i<8;i++){
                  sitemlinks[i]="http://www.mingrisoft.com";
            }
            for (i=0;i<sitems.length;i++){           //通过循环添加各菜单项
                  if (document.all) {
                        document.write('<tr><td  bgcolor=\'#EBFAFF\'  style=\'cursor:hand;\'  onclick="window.location.href=\''+sitemlinks[i]+'\'"
onmouseover="this.bgColor=\'#FFFDCD\'" onmouseout="this.bgColor=\'#EBFAFF\'">'+sitems[i]+'</td></tr>');
```

```
                    }else if (document.layers){
                        document.write('<tr><td bgcolor="white"><a href="'+sitemlinks[i]+'">'+sitems[i]+'</a></td></tr>');
                    }
                }
            </script>
        </table>
    </div>
</div>
```

（2）编写自定义 JavaScript 方法 mouseover()，用于在鼠标经过时显示导航菜单，关键代码如下：

```
function mouseover(divId){
    document.getElementById(divId).style.display="block";              //显示菜单
}
```

（3）编写自定义的 JavaScript 方法 mouseout()，用于在鼠标移出时隐藏导航菜单，关键代码如下：

```
function mouseout(divId){
    document.getElementById(divId).style.display="none";               //隐藏菜单
}
```

秘笈心法

本实例是应用 JavaScript 添加导航菜单的每一个菜单项。在 JavaScript 中，应用循环数组来动态添加多个菜单项，而不必在网页中添加大量导航内容的 HTML 代码，从而减少了 HTML 代码占用的行数。

第18章

表单的应用

- 文本框/编辑框/隐藏域组件
- 下拉列表与菜单的应用
- 单选按钮
- 复选框
- 密码域
- 表单的应用

18.1　文本框/编辑框/隐藏域组件

实例 470　获取文本框/编辑框/隐藏域的值　初级

光盘位置：光盘\MR\18\470　实用指数：★★★

实例说明

用户在浏览网页时，经常需要填写动态表单。例如，在互联网上发表自己的文章时，需要填写作者名称、文章标题以及文章内容。当用户提交表单时，程序需要获取表单内容，并对表单内容进行验证或存储。本实例将介绍如何获取表单中的文本框、编辑框以及隐藏域的值，程序运行结果如图 18.1 所示。

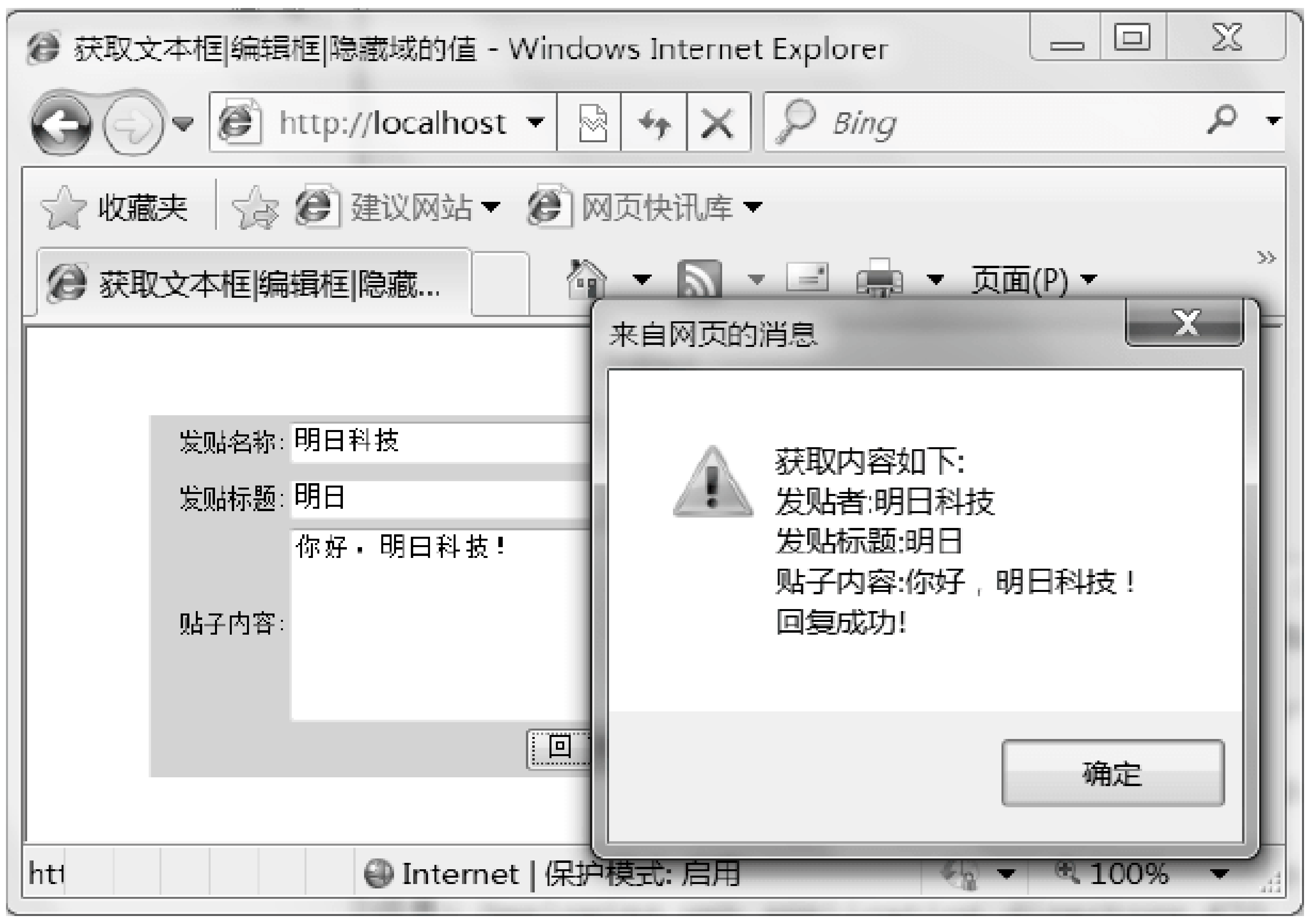


图 18.1　获取到的文本框、编辑框、隐藏域的值

关键技术

本实例应用的是 JavaScript 脚本的 document 对象从客户端获取相应的信息，再调用 windows 对象的 alert()方法输出动态显示获取的信息。如表 18.1 所示为 document 对象常用的属性和方法。

表 18.1　document 对象常用的属性和方法

属　　性	描　　述
document.forms[]	表示文档中所有表单元素的数组，[]内为索引值指向数组
document.images[]	表示使用<img>标记插入到文档中的图像数组
document.title	表示获取网页的标题文本
document.write()	指定字符串，写入到一个或多个 jsp 表达式中
document.writeln()	在输出参数之后附加一个换行符

设计过程

（1）创建 index.jsp 文件，在页面编写回复表单的相关元素和隐藏域，关键代码如下：

```
<body style="font-size:12px">
<table width="50%"   border="0" align="center" cellpadding="0" cellspacing="0">
  <tr>
    <td align="left" valign="top"><table width="95%"   border="0" align="center" cellpadding="0" cellspacing="0">
      <tr>
```

```
            <td height="22" align="center">回复帖子</td>
        </tr>
    </table>
        <br>
        <table width="500" border="0" align="center" cellpadding="0" cellspacing="0" bgcolor="#d3d3d3">
            <form name="form1" onSubmit="Mycheck()" >
                <tr>
                    <td height="28" align="right">发帖名称:</td>
                    <td height="28"><input name="发帖者" type="text" class="textbox" id="发帖者"></td>
                </tr>
                <tr>
                    <td height="28" align="right">发帖标题:</td>
                    <td height="28"><input name="发帖标题" type="text" id="发帖标题" class="textbox"></td>
                </tr>
                <tr>
                    <td height="22" align="right">帖子内容:</td>
                    <td height="22"><textarea name="帖子内容" cols="45" rows="6" id="帖子内容"></textarea></td>
                </tr>
                <tr>
                    <td height="28" colspan="2" align="center"><input name="隐藏域" type="hidden" id="隐藏域" value="回复成功!">
                    <input name="add" type="submit" class="button" id="add" value="回  复">
                                                                                             
            <input type="reset" name="Submit2" value="重  置" class="button"></td>
                </tr>
            </form>
    </table></td>
  </tr>
</table>
```

（2）利用 JavaScript 获取编辑框、文本框，以及文本域和隐藏域的值，关键代码如下：

```
<script type="text/javascript">
function Mycheck(){
   var checkstr="获取内容如下:\n";
   if (document.form1.发帖者.value != ""){
        checkstr+="发帖者:"+document.form1.发帖者.value+"\n";
   }
   if (document.form1.发帖标题.value != ""){
        checkstr+="发帖标题:"+document.form1.发帖标题.value+"\n";
   }
   if (document.form1.帖子内容.value != ""){
        checkstr+="帖子内容:"+document.form1.帖子内容.value+"\n";
   }
   if (document.form1.隐藏域.value != ""){
        checkstr+=document.form1.隐藏域.value;
   }
   if (checkstr != ""){
      alert(checkstr);
 return false;
   }
   else return
      return true;
}
</script>
```

秘笈心法

对象的一个最重要的特性是 write()方法，应用它可以从 JavaScript 程序中动态生成网页内容。

实例 471	自动预算 光盘位置：光盘\MR\18\471	初级 实用指数：★★★

实例说明

在实际项目开发中，为了快速开发程序并保证计算的准确性，可以利用 JavaScript 脚本实现自动预算。运行

本实例，在表单中填写“保险金额”、“保险日期”和“保险年限”后会自动计算用户如果违约应赔偿的金额，也就是“违约金”，如图 18.2 所示。



图 18.2　自动预算

关键技术

本实例主要是应用 JavaScript 的 document 对象来获取文本框和隐藏域的值而进行计算的，计算出结果的数值赋值给相应的文本框，在文本框中立即就能显示出违约金额。

设计过程

（1）自定义 JavaScript 函数，首先获取用户在文本框中的值，根据保险年限和隐藏域 value 属性的值相乘计算出违约金，关键代码如下：

```
<script type="text/javascript">
function autocount(){
   var amount,price,money;
   amount=document.form1.txt_year.value;
   price=document.form1.txt_by.value;
   money=amount*price;
   document.form1.txt_bail.value=money;
}
</script>
```

（2）在页面的保险年限使用 onBlur()事件中调用自定义的 JavaScript 函数 autocount()，关键代码如下：

```
<body style="font-size:12px">
        <form name="form1" >
        <div align="center" style="font-size: large" >保险合同金额预算</div>
        <table align="center" bgcolor="d3d3d3">
          <tr>
            <td   height="22" align="right">保险金额:</td>
            <td height="22" align="left"><input name="txt_money" type="text" id="txt_money2"></td>
          </tr>
          <tr>
            <td height="22" align="right">保险日期:</td>
            <td height="22" align="left"><input name="txt_date" type="text" id="txt_date"></td>
          </tr>
          <tr>
            <td height="22" align="right">保险年限:</td>
            <td height="22" align="left"><input name="txt_year" type="text" id="txt_year" size="10" onBlur="autocount()">
          年
            <input name="txt_by" type="hidden" id="txt_by" value="2000"></td>
          </tr>
          <tr>
            <td height="22" align="right">违约金:</td>
            <td height="22" align="left"><input name="txt_bail" type="text" id="txt_bail" readonly="true"></td>
          </tr>
          <tr>
            <td   height="32" align="right"> </td>
```

```
        <td height="22" align="left" colspan="2"><input type="submit" name="Submit3" value="保险合同 ">
        <input type="reset" name="Submit4" value="重新预算"></td>
        </tr>
        </table>
    </form>
```

秘笈心法

本实例用到了 onBlur 事件，它表示在文本框失去焦点时触发的事件。

实例 472	设置文本框为只读属性 光盘位置：光盘\MR\18\472	初级 实用指数：★★★

实例说明

网页中有一部分信息是不允许浏览者进行任何修改的，即用户只有浏览的权限，而没有修改的权限。这可以通过将文本字段设置为只读属性来实现。运行本实例，将鼠标或键盘放在“商品名称”处就会弹出提示信息，如图 18.3 所示。

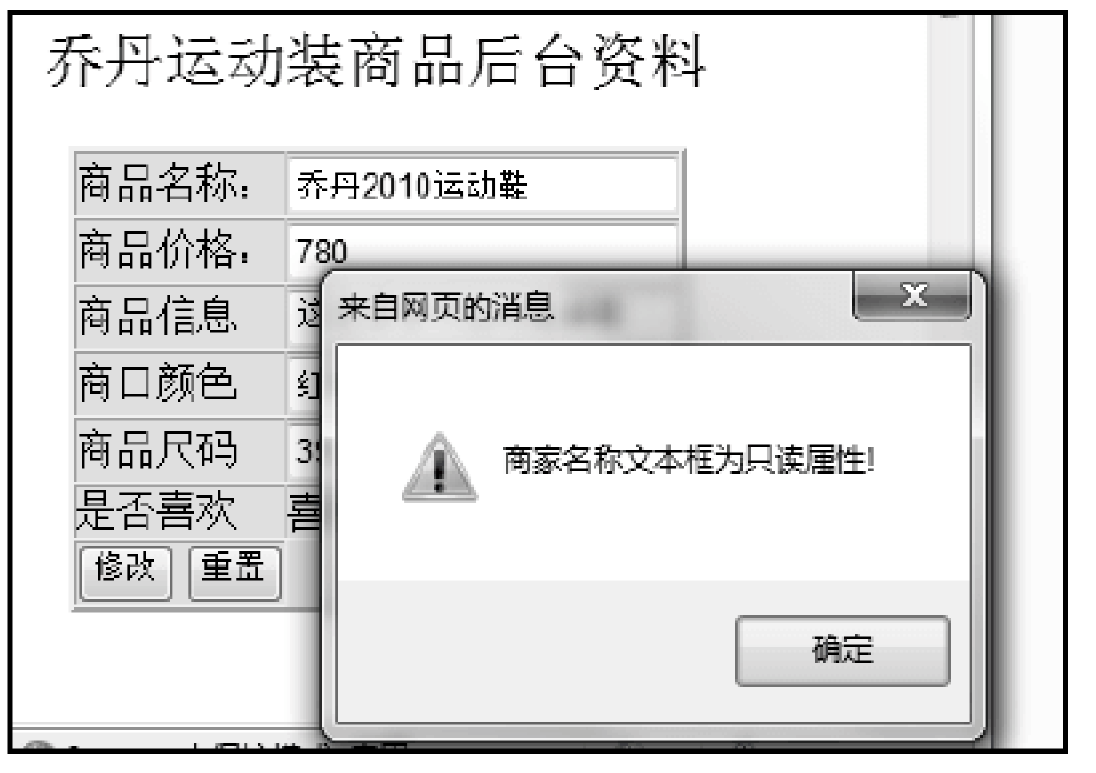


图 18.3　设置文本框为只读属性

关键技术

实现本实例首先需要将文本字段设置为只读属性，然后通过设置文本字段的单击事件来实现当用户单击已经设置为只读属性的文本字段时，弹出一个提示对话框，提示用户不能进行修改。

将文本字段设置为只读属性有两种方法，一种是应用文本字段的 readonly 属性实现，另一种是应用文本字段的 disabled 属性实现。

设计过程

（1）创建 main.jsp 页面文件，在“商品名称”的<input>标记中设置文本框的 readonly 属性值为 true，并在 onKeyDown 事件和 onMouseup 事件中调用自定义的 JavaScript 函数 Mycheck()，关键代码如下：

```
<Td>
    商品名称：
</Td>
<td>
    <input type="text" name="mname" id="mname" value="乔丹 2010 运动鞋"
        readonly="true" onKeyDown="Mycheck()" onMouseup="Mycheck()" />
</td>
</Tr>
```

（2）自定义的 JavaScript 函数 Mycheck()，关键代码如下：

```
<script type="text/javascript">
function Mycheck(){
   if (document.form1.mname.readOnly){
      alert("商家名称文本框为只读属性!");return false;
   }
   else return
      return true;
}
</script>
```

秘笈心法

文本框的 readonly 属性和 disabled 属性很相似，都用于禁止修改文本框的内容。但它们是有区别的，readonly 属性只对文本框和文本域有效；而 disabled 属性是对表单的所有元素有效，并且在提交表单时，设置为 disabled 属性的表单元素在提交后无法获取到值，而设置为 readonly 的表单元素是可以获取到值的。

实例 473　限制文本域字符个数

初级

光盘位置：光盘\MR\18\473

实用指数：★★★

实例说明

在网站的论坛或留言簿中，通常都是在多行文本域中输入留言的内容，而多行文本域不具备限制用户输入最多字符个数的功能，因此要想控制多行文本域输入的字符个数，必须由程序员自己编写代码。运行本实例，当用户输入的字符超过限制的字符个数时，则弹出提示信息对话框，如图 18.4 所示。

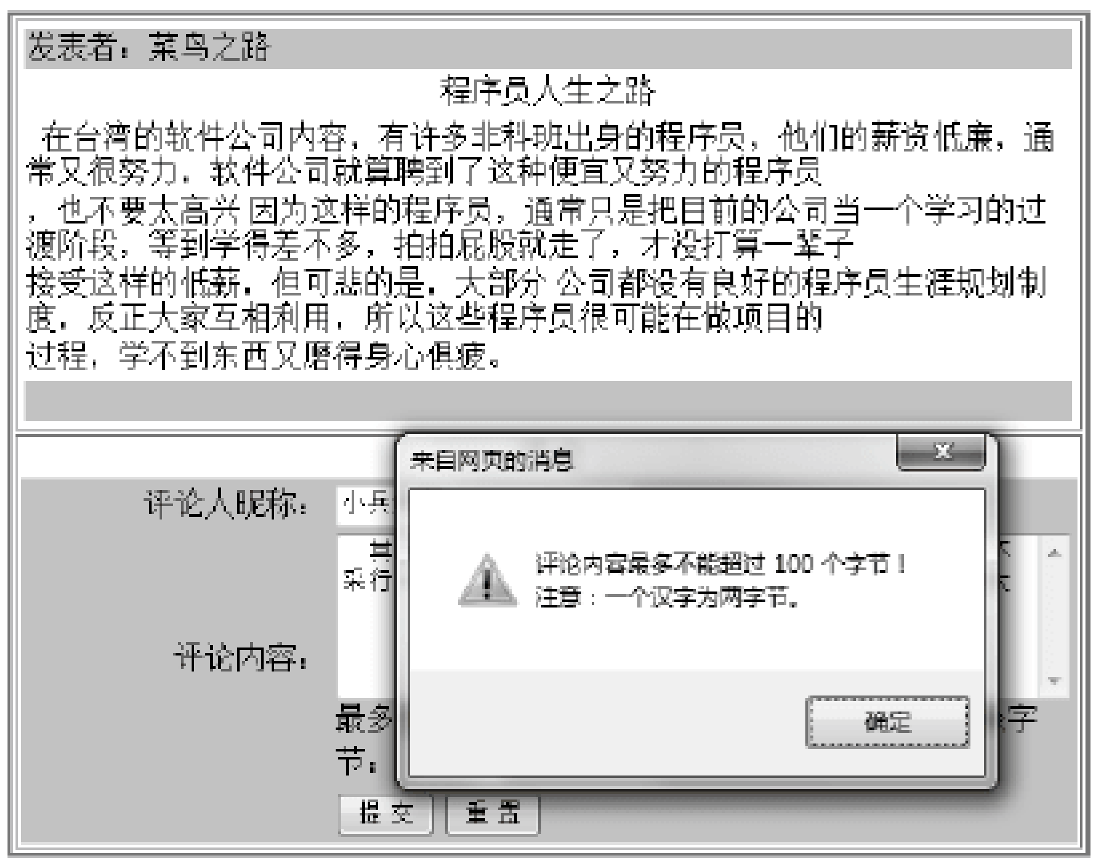


图 18.4　限制多行文本域输入的字符个数

关键技术

实现限制多行文本域输入的字符个数的功能，关键是在其 onKeyDown 事件和 onKeyUp 事件中调用自定义的 JavaScript 函数 CountStrByte()来限制输入的最多字符数并计算已输入字节数与剩余字节数。下面介绍 onKeyDown 事件和 onKeyUp 事件。

- ❑ onKeyDown 事件：访问者按下键盘上一个或几个键的事件。
- ❑ onKeyUp 事件：访问者按下键盘上一个或几个键后释放的事件。

设计过程

（1）首先创建 index.jsp 页面，在文本域<textarea>标记的 onKeyDown 事件和 onKeyUp 事件中分别调用自定义 JavaScript 脚本函数 CountStrByte()，关键代码如下：

```
<td height="22" align="left">
    <textarea name="评论内容" cols="45" rows="5" id="评论内容"
        onKeyDown="CountStrByte(this.form.评论内容,this.form.total,this.form.used,this.form.remain);"
    onKeyUp=CountStrByte(this.form.评论内容, this.form.total, this.form.used,
        this.form.remain);;></textarea>
        <br />
        最多允许
        <input name="total" type="text" disabled class="textbox"
            id="total" value="100" size="3">
        个字节  已用字节：
        <input name="used" type="text" disabled class="textbox"
            id="used" value="0" size="3">
        剩余字节：
        <input name="remain" type="text" disabled class="textbox"
            id="remain" value="100" size="3">
    </td>
```

（2）自定义 JavaScript 函数，此函数首先是使用 if…else 条件判断语句和 for 循环语句判断文本域输入的字符个数是否超过允许的最多字符个数，如果输入的字符个数超过允许字符限制，则使用 alert()方法弹出一个提示对话框，并立即退出循环，关键代码如下：

```
<SCRIPT language=JavaScript>
<!--
var LastCount =0;
function CountStrByte(Message,Total,Used,Remain){            //字节统计
 var ByteCount = 0;
 var StrValue = Message.value;
 var StrLength = Message.value.length;
 var MaxValue = Total.value;

 if(LastCount != StrLength) {                                 //在此判断，减少循环次数
 for (i=0;i<StrLength;i++){
        ByteCount    = (StrValue.charCodeAt(i)<=256) ? ByteCount + 1 : ByteCount + 2;
        if (ByteCount>MaxValue) {
                Message.value = StrValue.substring(0,i);
                alert("评论内容最多不能超过 " +MaxValue+ " 个字节！\n 注意：一个汉字为两个字节。");
            ByteCount = MaxValue;
            break;
        }
 }
    Used.value = ByteCount;
    Remain.value = MaxValue - ByteCount;
    LastCount = StrLength;
 }
}
//-->
</SCRIPT>
```

秘笈心法

在计算字符数时，需要判断输入的字符是汉字还是英文字母或数字，因为一个汉字占两个字节。另外，onKeyDown 事件和 onKeyUp 事件只适合于 Internet Explorer 4.0 及 4.0 以上浏览器。

实例 474	自动选择文本框和编辑框的文字 光盘位置：光盘\MR\18\474	初级 实用指数：★★★

实例说明

在实际开发中，经常要对大量的文字信息进行编辑，尤其是更新网页信息时，为了节省对文字编辑的时间或者引起用户的注意，可以自动选择编辑区域内的文字内容，以方便用户进行操作。运行本实例，当单击文本框或编辑框（即文本框或编辑获得焦点）时，文本框和编辑框中的内容将自动全部选中，如图 18.5 所示。

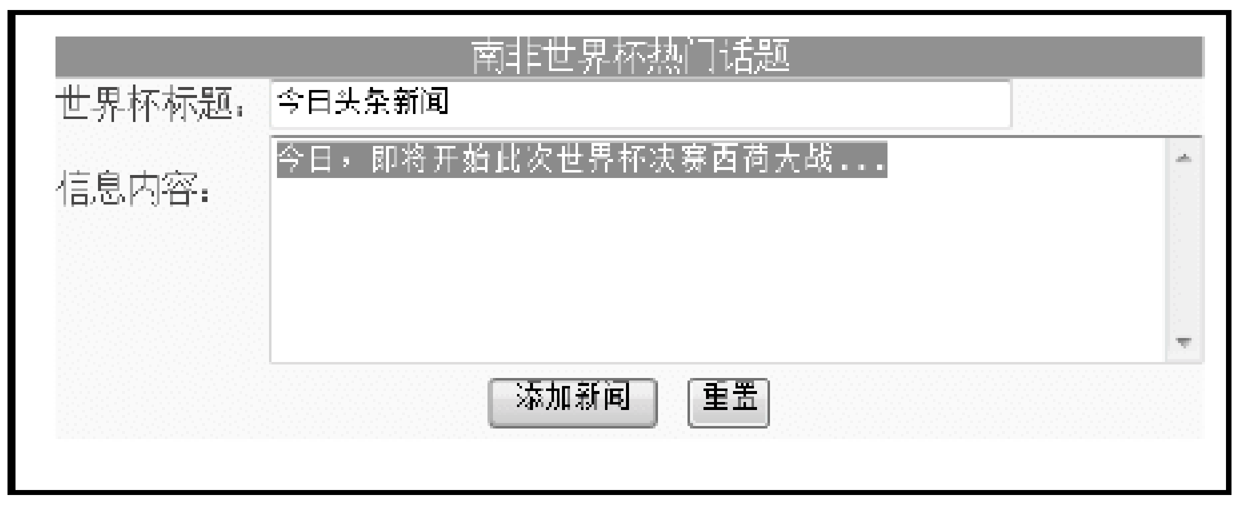


图 18.5　自动获取文本框和编辑框中的内容

关键技术

实现本实例，首先在自定义的 JavaScript 函数中，通过 document 对象获取文本框和编辑框对象，然后应用其 focus()函数判断文本框或编辑框是否获得焦点，如果已经获得焦点，则调用其 select()属性即可实现文字的选中状态。

设计过程

（1）首先创建 index.jsp 页面，编写自动选择文本框和编辑框内容的 JavaScript 自定义函数，关键代码如下：

```
<script type="text/javascript">
function select_txt(){
   if (document.form1.title.focus){
        document.form1.title.select();}
}
function select_txtarea(){
   if (document.form1.content.focus){
        document.form1.content.select();}
}
</script>
```

（2）在<input>标记和编辑框<textarea>标记的 onClick 事件中调用自定义的 JavaScript 函数，关键代码如下：

```
<tr>
   <td><div align="right"><font color="#214994">世界杯标题：</font></div></td>
   <td align="left"><input name="title" type="text" size="50" value="今日头条新闻" onClick="select_txt()"></td>
</tr>
<tr>
   <td><div align="left"><font color="#214994">信息内容：</font></div></td>
   <td rowspan="2"><textarea name="content" cols="50" rows="6" onClick="select_txtarea()">今日，即将开始此次世界杯决赛西荷大战...</textarea></td>
</tr>
```

秘笈心法

在本实例中用到的 focus()函数是 JavaScript 中的一个聚焦函数，一般在表单注册时，会应用 JavaScript 的 focus()函数使表单元素获得焦点。

实例 475

按 Enter 键时自动切换焦点

光盘位置：光盘\MR\18\475

初级

实用指数：★★★

实例说明

在设置表单时，为了方便用户填写表单，可以设置按下 Enter 键自动切换到下一行控件的焦点，而不是直接提交表单。运行本实例，当用户填写完一项信息后，按下 Enter 键时焦点自动切换到下一个文本框中，程序运行结果如图 18.6 所示。



图 18.6　按下 Enter 键时自动切换下一行

关键技术

本实例主要使用 JavaScript 的 Event 对象的 keyCode 只读属性。Event 对象的确切属性依赖于发生的事件。下面给出 Event 对象的常用属性，如表 18.2 所示。

表 18.2　Event 对象的常用属性

属　性	描　述
keyCode	表示按下按键的数字代号
Type	事件的名称
ctrlKey	值为 true 表示按下 Ctrl 键，值为 false 表示没有按下 Ctrl 键
Button	对于特定的鼠标事件，表示按下的鼠标按钮
altKey	值为 true 表示按下 Alt 键，值为 false 表示没有按下 Alt 键

设计过程

（1）创建 index.jsp 页面文件，在<input>标记中使用 onKeyPress 事件调用自定义的 JavaScript 的 enter()函数，关键代码如下：

```
<form name="form1">
   <tr bgcolor="#ffdeab"><td height="27" colspan="2" align="left">会员注册</td></tr>
   <tr>
     <td width="150" height="22" align="right">会员名称:</td>
     <td width="350" height="22"><input name="会员名称" type="text" class="textfiled" id="会员名称" size="25" onKeyPress="enter(form1.密码)"
/></td>
   </tr>
   <tr >
     <td height="22" align="right">密 码:</td>
     <td height="22"><input name="密码" type="password" class="textfiled" id="密码" size="26" onKeyPress="enter(form1.QQ 号码)" /></td>
   </tr>
   <tr >
     <td height="22" align="right">QQ 号码:</td>
     <td height="22"><input name="QQ 号码" type="text" class="textfiled" id="QQ 号码" size="25" onKeyPress="enter(form1.联系方式)" /></td>
   </tr>
```

（2）按下 Enter 键时自动切换焦点，关键代码如下：

```
<script type="text/javascript">
function enter(str){
  if (event.keyCode == 13){
    str.focus();}
}
</script>
```

秘笈心法

Event 代表事件的状态，它表示对象在事件发生过程中才有效，如 Event 对象的元素、鼠标的位置及状态、按下的键等。

18.2　下拉列表与菜单的应用

在开发网络程序时，对下拉列表的应用非常广泛，通过下拉列表用户不仅可以查看多种选项，在页面中还可以使用多个下拉列表制作多级菜单，为用户提供更多信息。本节将介绍下拉列表的应用。

实例 476	获取下拉列表、菜单的值 光盘位置：光盘\MR\18\476	初级 实用指数：★★★

实例说明

在设计网页时，可以使用下拉列表或菜单提供多个选项，不仅为用户提供方便，程序人员还可以根据获取到的下拉列表或菜单将用户提交的信息进行分类存储，保证对数据库信息的有效维护。运行本实例，填写相应的下拉列表和文本框的值，单击“激活”按钮，在弹出的窗口中将显示用户在下拉列表选择的选项名称，如图 18.7 所示。

关键技术

本实例主要应用 JavaScript 的 document 对象获取下拉列表、菜单的值，并使用 alert()方法弹出一个提示对话框，显示获取的相应内容，语法格式如下：

```
val1=document.form1.gameadd.value;
```

其中，参数 form1 为表单的 name 值，gameadd 为下拉列表的 name 值，value 为具体输入在下拉列表或文本

框中的内容。



图 18.7　获取下拉列表、菜单的值

设计过程

（1）创建 index.jsp 页面文件，在 form 表单中添加 onSubmit 事件并调用 check()方法，关键代码如下：

```
<form name="form1" onSubmit="check()" >
```

（2）编写获取下拉列表、菜单值的 JavaScript 脚本的 check()函数，关键代码如下：

```
<script type="text/javascript">
function check(){
   var val1,val2,val3;
   val1=document.form1.gameadd.value;
   val2=document.form1.line.value;
   val3=document.form1.linespeed.value;
   val4=document.form1.gamename.value;
   val5=document.form1.gameid.value;
   alert("获取下拉列表值:\n 服务器:"+val1+"\n 线路:"+val2+"\n 接网速度:"+val3+"\n 游戏账号:"+val4+"\n 角色 ID"+val5);
}
</script>
```

秘笈心法

onSubmit()事件处理函数可以用来在提交过程前验证表单，这仅限于使用提交按钮和图片按钮，如果使用的是 submit()方法，不会触发 submit 事件，所以所有的验证过程应该在调用它之前完成。

实例 477	遍历多选下拉列表 光盘位置：光盘\MR\18\477	初级 实用指数：★★★

实例说明

在使用下拉列表时，用户并不局限于只能选择一个选项，还可以在下拉列表中选择多个选项。运行本实例，用户可以按住 Ctrl 键并使用鼠标左键单击所要的选项，或者按住 Shift 键再单击鼠标左键选择所要的区域，实现在下拉列表中选择多个选项，选择的内容将显示在编辑框中，程序的运行结果如图 18.8 所示。

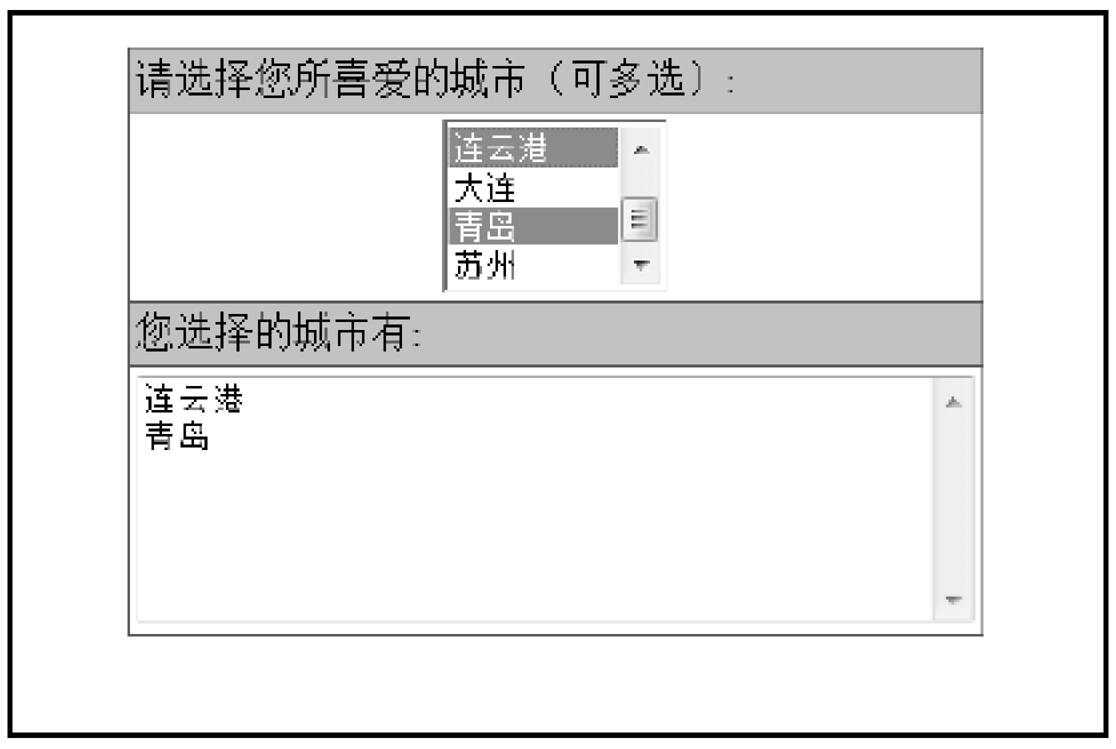


图 18.8　多选下拉列表的选择

关键技术

实现本实例，首先设置下拉列表的 multiple 属性，即允许用户选择多个选项，然后应用 JavaScript 脚本获取下拉列表的 length 属性值，再使用 for 循环判断 select 元素对象的 selected 属性值，如果值为 true（即此选项已被选中），则将此选项 value 值赋给一个指定变量，并将此变量值赋给页面中的编辑框。

设计过程

（1）创建 index.jsp 页面文件，在页面中添加下拉列表并设置 multiple 属性，并在下拉列表框的 onChange 事件中调用自定义的 JavaScript 函数，关键代码如下：

```
<tr>
<td   align="center" bgcolor="#FFFFFF">
<select name="mcusep" size="4" multiple class="textfiled" onchange="moreselect()">
        <option value="长春">车之城长春</option>
        <option value="淄博">淄博</option>
        <option value="连云港">连云港</option>
        <option value="大连">大连</option>
        <option value="青岛">青岛</option>
        <option value="苏州">苏州</option>
</select>
</td>
</tr>
```

（2）自定义 JavaScript 脚本函数 moreselect()，关键代码如下：

```
<script language="javascript">
function moreselect(){
    var orderstring="";
    var n=document.form1.mcusep.length;
    for (i=0;i<n;++i){
        if (document.form1.mcusep.options[i].selected){
            orderstring+=document.form1.mcusep.options[i].value+"\n";
        }
    }
    document.form1.textarea.value=orderstring;
}
</script>
```

秘笈心法

在自定义的 JavaScript 函数中，获取到 select 元素对象后，调用其 options 属性返回的是下拉列表所有选项的数组，然后在循环中应用 options[i]即可获取下拉列表中的每一个选项。

实例 478	在下拉列表中进行多选择移除 光盘位置：光盘\MR\18\478	初级 实用指数：★★★

实例说明

在开发动态网站程序时，经常会遇到将下拉列表中的选项进行多选移除或者多选移入。运行本实例，选择喜欢的书籍，单击“》”按钮，此时将显示在选项 2 的下拉列表中，如果选择选项 2 下拉列表的选项，并单击“《”按钮，此时将出现在选项 1 的下拉列表中，运行结果如图 18.9 所示。

关键技术

本实例主要应用 JavaScript 的 while 循环语句，判断如果 select 元素的 selectedIndex 属性值不为-1，则获取下拉列表中选中项的索引值和对应的文本，然后使用 select 元素对象的 add()方法将此选项添加到另一个下拉列表中，并应用 select 元素对象的 remove()方法移除当前下拉列表中的此选项。

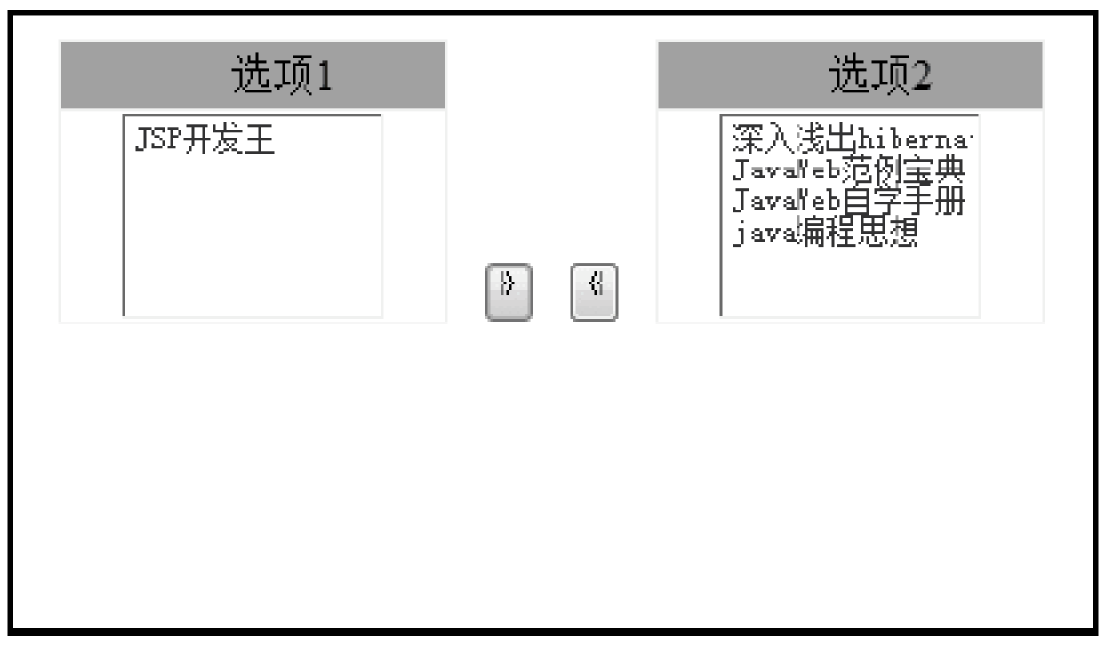


图 18.9　下拉列表中进行多选移动

设计过程

（1）创建 index.jsp 页面，在页面中分别添加两个下拉列表和普通按钮，在按钮的 onClick 鼠标单击事件中调用自定义的 JavaScript 脚本函数，关键代码如下：

```
<select name="sel_place1" size="6" multiple class="wenbenkuang" id="sel_place1" style="width:100px " >
          <option value="sel1">JavaWeb 自学手册</option>
          <option value="sel2">java 编程思想</option>
          <option value="sel3">深入浅出 hibernate</option>
          <option value="sel4">JSP 开发王</option>
          <option value="sel5">JavaWeb 范例宝典</option>
</select>
<input name="sure2" type="button" id="sure2" onClick="mulselect(document.form1.sel_place1,document.form1.sel_place2);" value="》" align="center" >
   
   <input name="sure1" type="button" id="sure1"   onClick="mulselect(document.form1.sel_place2,document.form1.sel_place1);" value="《">
```

（2）自定义的 JavaScript 函数，关键代码如下：

```
<script language="javascript">
function mulselect(n1,n2){
  while(n1.selectedIndex!=-1){
       var indx=n1.selectedIndex;
       var t=n1.options[indx].text;
       n2.options.add(new Option(t));
       n1.remove(indx);
  }
}
</script>
```

秘笈心法

Select 下拉列表的 selectedIndex 属性可设置或返回下拉列表中选项的索引号，如果所选的内容为空则返回-1，否则返回索引值。

实例 479	将数组中的数据添加到下拉菜单中 光盘位置：光盘\MR\18\479	初级 实用指数：★★★

实例说明

在开发动态网站时，可以将一些固定不变的数据存储在数组中，然后将数组中的书籍显示到下拉菜单中以供用户选择，这样可以加快程序的运行速度，方便用户浏览网页。运行本实例，单击下拉菜单的控制按钮，将显示出该下拉菜单中的全部选项，运行结果如图 18.10 所示。

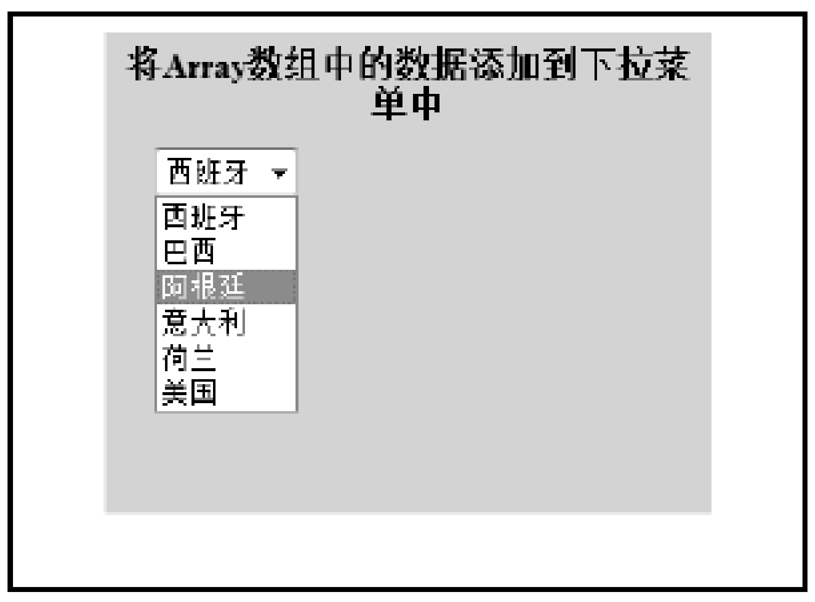


图 18.10　将数组中的数据添加到下拉菜单中

关键技术

本实例主要使用 JavaScript 的构造函数 Array()和 Option()。下面分别进行介绍。

（1）Array()构造函数和运算符 new 同样可以创建 Array 数组对象，并且可以使用多种方式创建数组。

（2）Option()构造函数可以动态地创建 Option 对象，语法格式如下：

```
new Option(text,value,defaultSelected,selected)
```

参数说明

❶ text：表示一个字符串，指定 Option 对象的 text 属性。

❷ value：表示一个字符串，指定 Option 对象的 value 属性。

❸ defaultSelected：为布尔值，指定 Option 对象的 defaultSelected 属性。

❹ selected：为布尔值，指定 Option 对象的 selected 属性。

设计过程

（1）在 JavaScript 脚本中定义一个数组，并为数组赋值，然后自定义一个 select()函数，将数组中的数组添加到表单的下拉菜单中，关键代码如下：

```
<script type="text/javascript">
var counts;
counts=0;
arr = new Array("西班牙","巴西","阿根廷","意大利","荷兰","美国");
counts=arr.length;
function select(){
   var i;
   for (i=0;i < counts; i++)  {
      document.form1.sele.options[i] = new Option(arr[i],i);
   }
}
</script>
```

（2）在下拉列表标记的 onFocus 事件中调用自定义的 JavaScript 函数 select()，关键代码如下：

```
<select name="sele" id="sel" onFocus="select()"></select>
```

秘笈心法

JavaScript 的构造函数本身没有类的概念，只有函数的概念，JavaScript 的类实际上也是一个 JavaScript 的函数，在这个特殊的函数中可以包含变量和其他 JavaScript 函数的引用。

实例 480　下拉菜单选择所要联机的网站

光盘位置：光盘\MR\18\480

初级

实用指数：★★★

实例说明

在进行网站程序设计时，大多数网站都设有友情链接这一版块，通常情况下，链接到一些相关及热门网站，浏览者可以通过单击网站名称或地址的超链接进入所要访问的网站。有时为了节约页面也可以通过下拉菜单的方法来实现该功能。运行本实例，当用户选择下拉菜单中所要访问的网站后，单击“进入”按钮可以立即链接到指定的网站，程序的运行结果如图 18.11 所示。



图 18.11　下拉菜单选择所要联机的网站

关键技术

本实例将下拉菜单中各选项的 value 属性设置为对应联机网站的访问，并在自定义 JavaScript 函数 gotoURL()中获取用户选择的下拉列表选项值，然后使用 window 对象的 location 属性指定链接的 URL 地址。

设计过程

（1）自定义 JavaScript 函数，此函数首先判断浏览者是否选择了“新窗口访问站点”，如果选中此复选框，

则使用 window 对象的 open()方法打开一个新窗口浏览页面，否则在本窗口中打开新页面，关键代码如下：

```
<script language="javascript">
<!--
function gotoURL(){
        var thebox=document.form1
        if(thebox.fig.checked){
                if(!window.newwindow){
                        newwindow=window.open("")
                        newwindow.location=thebox.example.options[thebox.example.selectedIndex].value
                }
        }
        else
                window.location=thebox.example.options[thebox.example.selectedIndex].value
}
//-->
</script>
```

（2）在<input>标记的 onClick 事件中调用自定义的 JavaScript 函数 gotoURL()，关键代码如下：

```
<form name="form1" >
        <input type="checkbox" name="fig" value="on">
        <span >新窗口访问站点</span><br>
        <select name="example" onchange="">
            <option value="http://www.sina.com">新浪</option>
            <option value="http://www.sohu.com">搜狐</option>
            <option value="http://www.taipingyang.com">太平洋电脑</option>
            <option value="http://www.tianya.com">天涯论坛</option>
            <option value="http://www.163.com">网易</option>
        </select>
        <input name="button" type="button" onclick="gotoURL()" value="进入">
</form>
```

秘笈心法

本实例使用 window 对象的 open()方法打开新窗口对象的引用，并通过它来操作新窗口中的元素。

实例 481	多级级联菜单 光盘位置：光盘\MR\18\481	初级 实用指数：★★★

实例说明

在开发购物网站时，经常需要对商品或物品进行分类，分类可以有二级或者多级。在设置页面时，可以使用多个下拉菜单分别显示不同级别的分类信息，即实现多级级联菜单。运行本实例，当选择商品的“第一级分类”下拉菜单时，商品的“第二级分类”下拉菜单的内容会随即发生变化，程序的运行结果如图 18.12 所示。



图 18.12　多级级联菜单

关键技术

本实例首先在<script>标记中定义两个数组，分别用于记录商品一级分类和二级分类的信息，然后自定义一个 JavaScript 函数，此函数的功能是当触发“第一级分类”下拉菜单的 onChange 事件时，先清空商品的“第二级分类”下拉菜单的选项内容，然后再将对应的信息装载到商品的“第二级分类”下拉菜单中。

设计过程

（1）创建 index.jsp 页面，在表单中添加下拉菜单并使用 onChange 事件调用自定义的 JavaScript 函数 rinfo()和 Menu()，关键代码如下：

```
<select  name="类别"  id="类别"  onChange="Menu(arr2[document.form1.类别.options[document.form1.类别.selectedIndex].text],document.form1.分类);">
        <option selected>水果</option>
        <option>豆制品</option>
        <option>厨房</option>
</select>
<select name="分类" id="分类">
        <option>苹果</option>
        <option>西瓜</option>
        <option>香蕉</option>
</select>
```

（2）编写 JavaScript 脚本实现多级级联菜单，关键代码如下：

```
<script type="text/javascript">
var arr2 = new Array(4);
arr2["水果"] = new Array("苹果","西瓜","香蕉");
arr2["豆制品"] = new Array("酸奶","鲜奶");
arr2["厨房"] = new Array("洗洁精","厨具");
function rinfo(classMenu){
   for (var i=0; i < classMenu.options.length; i++){
      classMenu.options[i]=null;
   }
}c
function Menu(classList,classMenu){
   rinfo(classMenu)
   for (var i=0; i < classList.length; i++){
         classMenu[i]=new Option(classList[i],classList[i]);
   }
}
</scrcipt>
```

秘笈心法

在 JavaScript 中清空下拉列表的所有选项，可以通过设置下拉列表的 option 对象为 null 来实现。

实例 482	分级下拉列表 光盘位置：光盘\MR\18\482	初级 实用指数：★★★

实例说明

在制作网页时，如果信息分类的内容很多，还可以在下拉列表中将其分级显示，使用户更清晰所查看的选项。运行本实例，可以看到下拉列表的选项是分级显示的，程序的运行结果如图 18.13 所示。



图 18.13　分级下拉列表

关键技术

本实例主要使用<optgroup>标记，并设置其为 label 属性。<optgroup>标记主要用于对 select 元素中的选项进行逻辑分组，在<optgroup>标记中指定的文本是非可选项，一般通过替换文本与可选项来区分。

设计过程

创建 index.jsp 页面，在表单处添加下拉列表的<optgroup>属性，实现分级下拉列表，关键代码如下：

```
<select name="book_class" id="book_class">
        <optgroup label="软件专业">
                <option>软件工程与开发</option>
                <option>信息工程业</option>
                <option>Java 软件工程师</option>
```

```
        </optgroup>
        <optgroup label="机电一体化">
                <option>机械工程</option>
                <option>汽修维护</option>
                <option>模具维修</option>
        </optgroup>
        <optgroup label="商务英语">
                <option>专业英语</option>
                <option>企业英语</option>
                <option>旅游英语</option>
        </optgroup>
</select>
```

秘笈心法

<optgroup>标签的 label 属性值会作为组合的标题（第一级）显示在菜单中，因此此项一定要指定，在 option 标签的 label 属性值中，通过显示组合名称，可以指定能够省略部分的次选项，如果省略了这个属性，在 option 范围内指定的内容就会没有改变，直接使用原有的选项。

18.3　单 选 按 钮

单选按钮组是由一组单选按钮组成的，并且在一个单选按钮组中，每次只能有一个单选按钮被选中，它也是 HTML 页面中常用的表单元素之一。下面通过几个具体实例介绍单选按钮组的应用。

实例 483	不提交表单获取单选按钮的值 光盘位置：光盘\MR\18\483	初级 实用指数：★★★

实例说明

在网站注册时，需要对填写的内容进行验证，如用户名是否被占用、密码位数是否不够安全、身份证和 Email 是否合法等。判断用户输入信息的合法性有两种实现方法：一种是在提交表单以前，应用 JavaScript 实现；另一种是在表单提交后通过 VBScript 实现，大多数情况下都采用前者来判断。运行本实例，判断用户输入的证件号码是否正确，需要根据前面选择的证件类型进行判断，因为不同证件的号码位数也不同，如身份证需要判断 15 或 18 位，而军官证需要判断 8～12 位，程序的运行结果如图 18.14 所示。

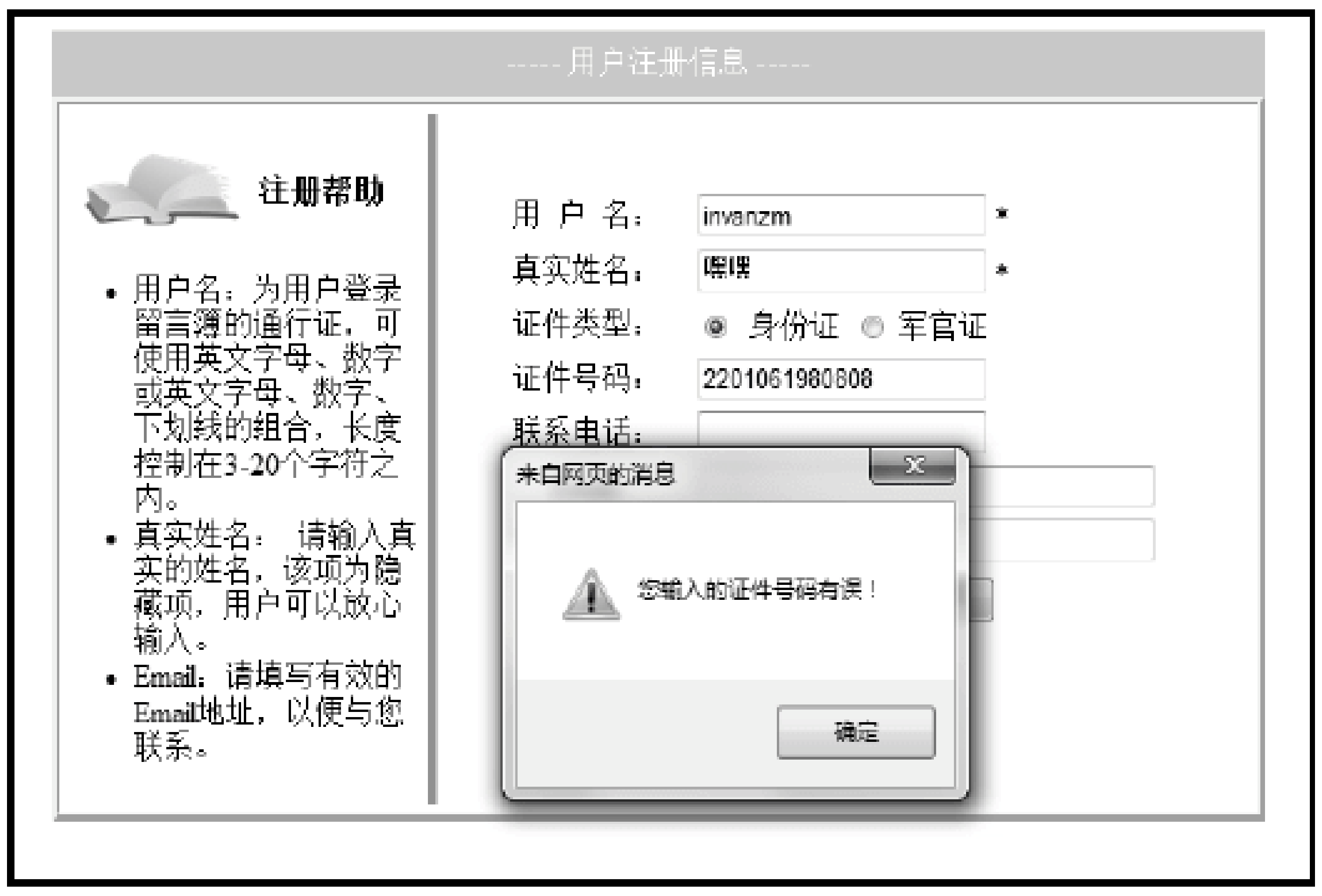


图 18.14　不提交表单获取单选按钮的值

关键技术

本实例主要是通过一组单选按钮组成的单选按钮组实现的，单选按钮组每次只能有一个单选按钮被选中。

单选按钮的属性和事件如表 18.3 所示。

表 18.3　单选按钮的属性和事件

属　　性	说　　明	事　　件	说　　明
Checked	表示单选按钮被选中	onBlur	失去焦点时的动作
Name	对象属性访问名称	onClick	单击鼠标时的动作
Type	对象类型	onFocus	获得焦点时的动作
Value	对象值		

设计过程

（1）创建 index.jsp 页面，在身份证和军官证的<input>标记上使用 onBlur 事件并调用自定义的 JavaScript 函数，关键代码如下：

```
<tr>
  <td height="28" align="center">证件类型：</td>
  <td>
            <input name="CardType" type="radio" class="noborder" value="身份证" checked>身份证 
      <input name="CardType" type="radio" class="noborder" value="军官证">军官证
</td>
</tr>
```

（2）编写实现不提交获取单选按钮的值，关键代码如下：

```
<script type="text/javascript">
function getVal(){
 var CardTypeValue;
 for (i=0;i<form1.CardType.length;i++){
      if (form1.CardType[i].checked){
            CardTypeValue=form1.CardType[i].value;
            break;
      }
 }
 if(CardTypeValue=="身份证"){
      if(form1.pcard.value.length!=15 && form1.pcard.value.length!=18){
            alert("您输入的证件号码有误！");form1.CardType.focus();return;
      }
 }else{
      if(CardTypeValue=="军官证"){
            if(form1.pcard.value.length!=8 && form1.pcard.value.length!=12){
                  alert("您输入的证件号码有误！");form1.CardType.focus();return;
            }
      }
 }
}
</script>
```

秘笈心法

在本实例中，验证证件类型时用到了 JavaScript 脚本；在选取用户输入的内容长度时，用下面的代码来获取：

```
Form1.pcard.value.lenght
```

实例 484	选中单选按钮后显示其他表单元素 光盘位置：光盘\MR\18\484	初级 实用指数：★★★

实例说明

在设计网站新闻信息添加页面时，根据选择的信息类型的不同，需要显示不同表单元素。运行本实例，选中“滚动条消息”单选按钮时，其下方显示“消息主题”文本框和“消息内容”编辑框，如图 18.15 所示；如

果选中“浮动条消息”单选按钮，在其下方增添一个“浮动条相关图片”文本框，如图 18.16 所示。

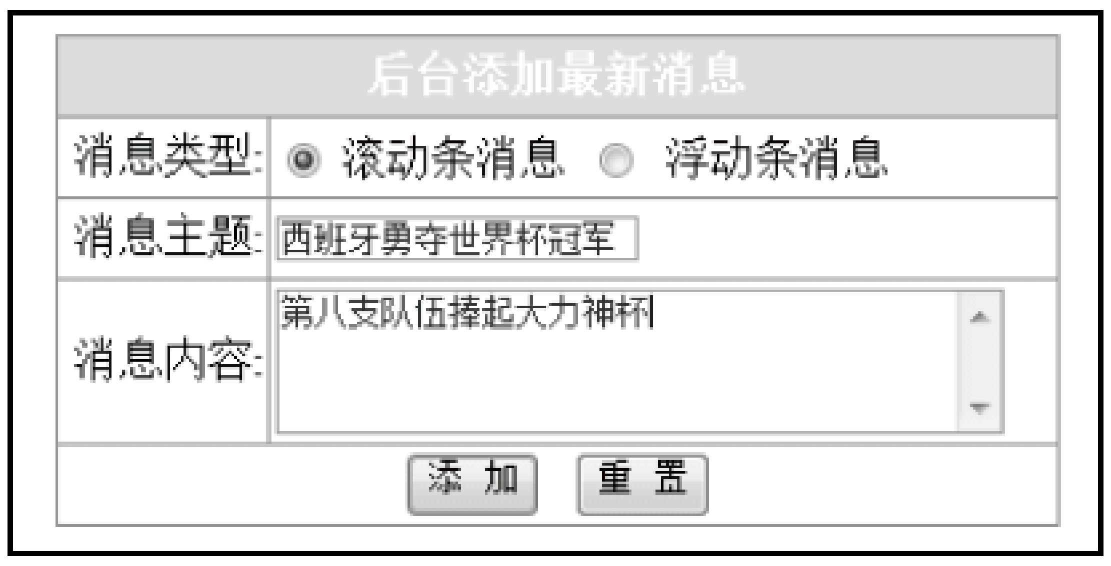


图 18.15　选中“滚动条消息”单选按钮

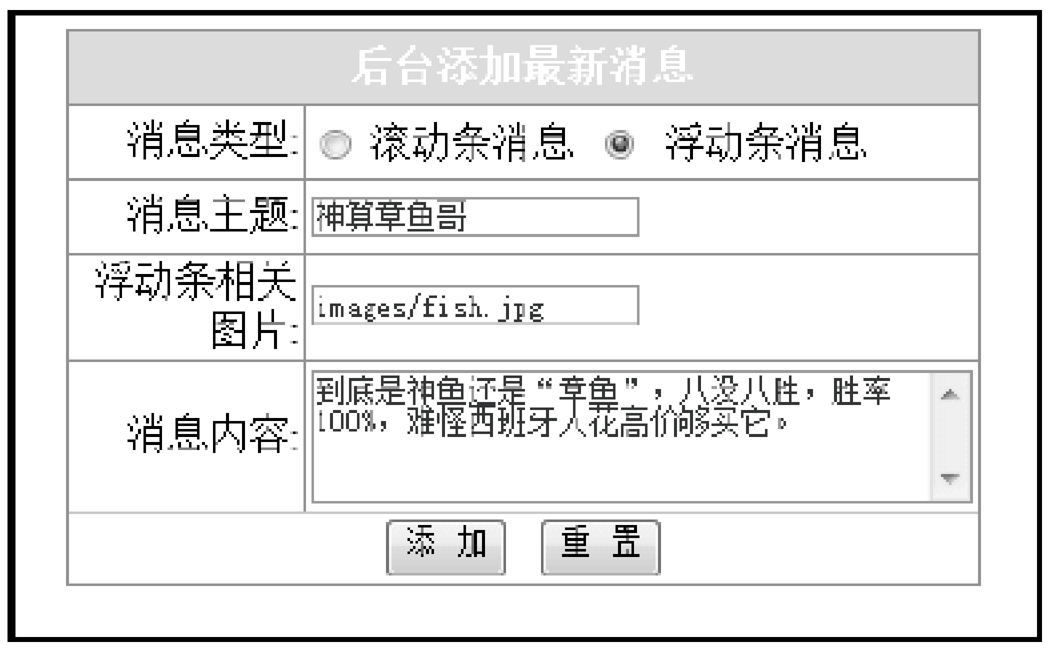


图 18.16　选中“浮动条消息”单选按钮

关键技术

本实例主要应用 JavaScript 来设置表格的 display 样式，以实现表格的显示和隐藏。通过设置表格的 id 属性，可以实现对表格的整体控制。通过将表格的 display 样式设置为 none，可以控制表格的隐藏；将表格的 display 样式设置为 block，可以显示表格。

设计过程

（1）创建 index.jsp 页面，设置“浮动条相关图片”的表格 id 属性值，默认状态下是不可见的，关键代码如下：

```
<tr bgcolor="#FFFFFF" id="img" style="display:none">
```

（2）在选中“滚动条消息”单选按钮时，触发其 onClick 事件，使“浮动条相关图片”文本框不可见，关键代码如下：

```
<td height="28"><input name="info" type="radio" value="1" onClick="img.style.display='none';" checked>滚动条消息
```

（3）在选中“浮动条消息”单选按钮时，触发其 onClick 事件，使“浮动条相关图片”文本框显示可见，关键代码如下：

```
<input type="radio" name="info" value="2" onClick="img.style.display='block';">浮动条消息
```

秘笈心法

display 属性用来确定页面元素是否显示和显示方式，它是一个不可继承的属性。display 属性在指定元素的显示类型时，可以把原本内联属性元素通过 display 指为块元素。

实例 485　通过单选按钮控制其他表单元素是否可用

初级

光盘位置：光盘\MR\18\485

实用指数：★★★

实例说明

在开发动态网站时，根据用户选择的选项不同，可以设置表单中的元素是否可用，以保证提交信息的准确性。运行本实例，选中“高级会员”单选按钮时，用户可以使用会员功能中的“时尚音乐平台”和“选择房间”下拉列表，如图 18.17 所示；选中“普通会员”单选按钮时，会员功能只有“昵称”一项可修改，其他两项均为灰色不可用状态，如图 18.18 所示。



图 18.17　选中“高级会员”单选按钮



图 18.18　选中“普通会员”单选按钮

关键技术

本实例主要应用 JavaScript 来设置下拉列表的 disabled 属性，当元素的 disabled 属性为 true 时，表示该元素不可用，并且此元素呈灰色显示状态。

设计过程

（1）创建 index.jsp 页面，设置“时尚音乐平台”和“选择房间”默认情况下是可用的，关键代码如下：

```
<tr>
  <td   height="22" align="right" bgcolor="#FFFFFF">时尚音乐平台:</td>
  <td   bgcolor="#FFFFFF" ><select name="t_class" id="t_class">
     <option>柴米油盐酱醋</option>
     <option>爱你一万年</option>
     <option>...更多</option>
  </select>（限高级会员）</td>
</tr>
<tr>
<td align="right" bgcolor="#FFFFFF" >选择房间</td>
<td bgcolor="#FFFFFF" ><select name="t_class1" id="t_class1" >
<option value="80 后时尚 K 歌" >80 后时尚 K 歌</option>
<option value="爱别离之屋" >爱别离之屋</option>
<option value="闲聊之吧">闲聊之吧</option>
</select><input type="button" value="go!" />(限高级会员)</td>
</tr>
```

（2）当选中“高级会员”单选按钮时，触发其 onClick 事件并刷新当前页面，关键代码如下：

```
<input name="t_type" type="radio" value="1" onClick="window.location='index.jsp';" checked>
```

（3）如果当前选中的是“普通会员”单选按钮，触发其 onClick 事件，设置下拉列表的 disabled 属性值为 true，关键代码如下：

```
<input type="radio" name="t_type" value="2" onClick="t_class.disabled='true';t_class1.disabled='true';">
```

秘笈心法

disabled 属性值为 true 时，说明禁用了这个元素，如果不想让浏览者操作这个元素，而又想获取到其值，可以设置只读属性 readonly 为 true，这样属性为只读，后台又能获取到值。

18.4 复 选 框

在设置网页时，根据用户需求，可以使用复选框为用户提供多个选项，这样用户针对某一问题时，可以选择一个以上的选项。复选框显示一个带有标识的小方格，当用户单击时在其中会显示一个选择标志，或者将已有标志取消。本节将介绍复选框的应用。

实例 486 只有一个复选框时控制复选框的全选或反选 初级

光盘位置：光盘\MR\18\486 实用指数：★★★

实例说明

本实例将实现只有一个复选框时控制复选框的全选或反选功能，程序的运行结果如图 18.19 所示。

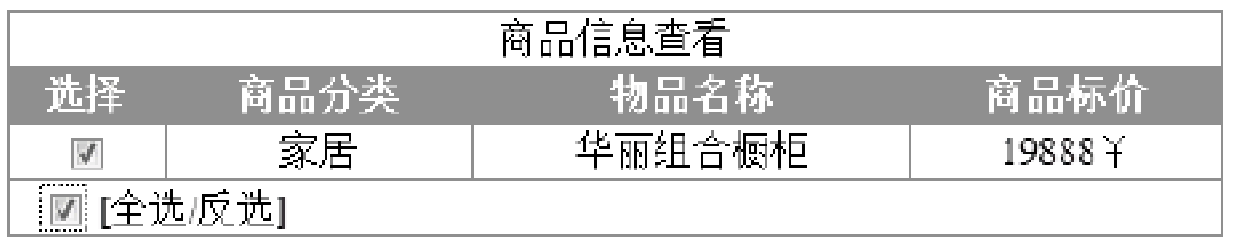


图 18.19 只有一个复选框时控制复选框的全选或反选

关键技术

本实例主要通过 JavaScript 的 if 条件判断语句确定复选框的数量是否大于 0，如果数值不大于 0，说明需要控制的复选框只有一个（即返回的 length 属性值为空），则可以直接设置复选框的 checked 属性值为 true 或 false。

设计过程

创建 index.jsp 页面，实现只有一个复选框的全选或反选的关键代码如下：

```
<script type="text/javascript">
function CheckAll(elementsA,elementsB){
        var len = elementsA;
        if(len.length > 0){
        }
        else{
                len.checked = true;
                if(elementsB.checked == false){
                        len.checked = false;
                }
        }
}
</script>
```

秘笈心法

在 JavaScript 中，获取到复选框对象后，判断其 checked 属性为 true 或 false，就可以知道当前这个复选框是否被选中。

18.5　密　码　域

在开发动态网站时，网站前台用户注册的页面、登录页面或者网站后台管理员登录的页面都要用到密码域，在页面中不仅可以获取到密码域的值，还可以对密码域进行设置使其内容更为安全。本节将介绍密码域的应用。

实例 487	让密码域更安全 光盘位置：光盘\MR\18\487	初级 实用指数：★★★

实例说明

对于网络应用的任何程序和网站，安全最为重要。而与其息息相关的便是密码域，虽然在密码域中已经将所输入的字符以掩码形式显示了，但是并没有真正保密，因为用户或浏览者可以通过复制该密码域的内容，并把复制的密码粘贴到其他文档上即可查看到用户输入的密码内容。为此可以将密码域的复制功能屏蔽，同时改变密码域的掩码符号。运行本实例，输入密码并选中输入的密码，单击鼠标右键时会看到复制的菜单项变为灰色即不可使用状态，复制和剪切也用不了，运行结果如图 18.20 所示。



图 18.20　让密码域更安全

关键技术

本实例主要是通过控制密码域的 oncopy、oncut、onpaste 事件来实现密码域的内容禁止复制的功能，并通过改变其 style 样式属性来实现改变密码域中掩码的样式。

设计过程

（1）创建 index.jsp 页面，在页面添加密码域，关键代码如下：

```
<input name="txt_passwd" type="password" class="textbox" id="txt_passwd" size="12" maxlength="50" >
```

（2）在页面添加禁止用户复制、粘贴、剪切密码操作，关键代码如下：

```
oncopy="return False" oncut="return False" onpaste="return False"
```

（3）改变密码域的掩码样式，将其 font-family 设置为 Wingdings，关键代码如下：

```
style="font-family:Wingdings;"
```

秘笈心法

需要注意的是，在修改掩码的样式时一定要选择 Windows 自带的字体样式，如果设置的字体样式不存在，密码会以原形显示。

实例 488　不提交表单自动检测密码域是否相同

光盘位置：光盘\MR\18\488

初级

实用指数：★★★

实例说明

在制作用户注册页面时，要求用户输入密码处都有“确认密码”一项，以确定用户输入密码是否准确。运行本实例，如果用户在“确认密码”文本框中输入的内容和“密码”文本框中输入的内容不同，则弹出提示框，要求用户再次确认输入的密码，运行结果如图 18.21 所示。



图 18.21　不提交表单自动检测密码域是否相同

关键技术

本实例主要应用 JavaScript 的 if 条件语句来判断两次输入的密码是否相同，如果不同，则调用 alert()方法弹出一个警告提示框。

设计过程

创建 index.jsp 页面文件，在确认密码域中的<input>标记实现不提交表单自动检测密码是否相同的关键代码如下：

```
<input name="pwd2" type="password" class="style1" id="pwd2" onBlur="if(this.value!=this.form.pwd1.value) alert('您两次输入的密码不一致！');" size="18">
```

秘笈心法

在本实例中，在“确认密码”文本框中应用了 onBlur 事件，应用它可以实现当输入确认密码并且确认密码的文本框失去焦点时，会调用 JavaScript 自定义函数来验证密码与确认密码是否一致。

18.6　表单的应用

表单是实现动态网页的一种主要外在形式，使用表单可以收集客户端提交的相关信息，是实现网站互动功能的重要组成部分。下面通过几个具体的实例介绍表单的应用。

实例 489　通过 JavaScript 控制表单的提交与重置

光盘位置：光盘\MR\18\489

初级　实用指数：★★★

实例说明

用户填写表单时，可以单击“提交”按钮提交表单，也可以直接按 Enter 键提交表单，同时还可以单击“重置”按钮将所填内容清空。为了防止用户因疏忽而按错键，可以使用 JavaScript 脚本来控制表单的提交与重置，程序的运行结果如图 18.22 所示。

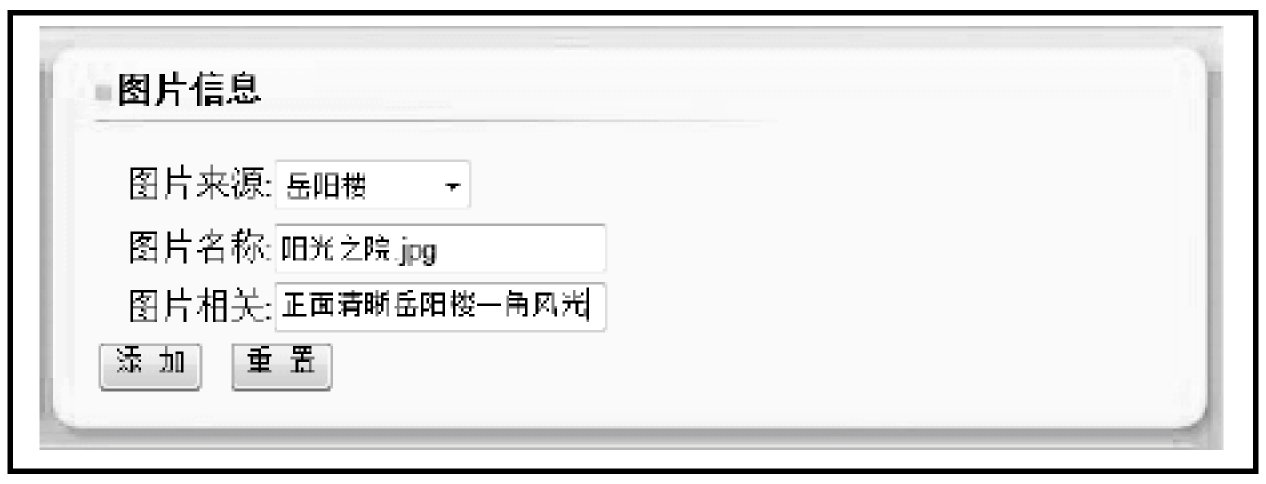


图 18.22　通过 JavaScript 实现表单的提交与重置

关键技术

本实例主要是通过 JavaScript 脚本来调用表单的 submit()提交方法和 reset()重置方法。页面中的按钮为普通按钮，即<input>标记的类型为 button。

设计过程

创建 index.jsp 页面文件，在“添加”和“重置”按钮处使用 onClick 事件，关键代码如下：

```
<td height="28" colspan="2" align="left">
<input name="add" type="button" class="button" id="add" value="添　加" onClick="document.form1.submit();">
<input type="button" name="Submit2" value="重　置" class="button" onClick="document.form1.reset();">
</td>
```

秘笈心法

重置表单在 Web 开发中已不再受追捧了，因为常常会有用户不小心单击“重置”按钮而不是“提交”按钮（因为“提交”和“重置”按钮往往靠在一起）。如果表单第一次载入时，在字段中已包含一些默认信息，那么“重置”按钮还有些作用，因为可恢复到初始值，但对载入时字段中没有任何信息的表单，最好避免使用“重置”按钮。

实例 490　带记忆功能的表单

光盘位置：光盘\MR\18\490

初级　实用指数：★★★

实例说明

在一般情况下，用户提交表单后，会跳转到另一个页面，同时表单中的内容也会清空。而有时为了简化操作步骤，需要保留历史信息，即当用户再返回原来页面时，还可以看到刚才所填写的信息。运行本实例，当用户刷新或者提交表单后再退回到原来页面时，“图书价格”和“出版社”文本框中的内容将保持不变，运行结果如图 18.23 所示。

关键技术

本实例主要通过在 CSS 样式定义中设置 behavior 确定对象的行为，并设置<meta>元信息标记中的两个属性

name 和 content 来实现保留历史信息的功能。<meta>标记用来在 HTML 文件中模拟 HTTP 协议的响应头报文，是实现元数据的主要标记，它可用于鉴别标注内容提要和关键字、设定页面字符集、刷新页面等。

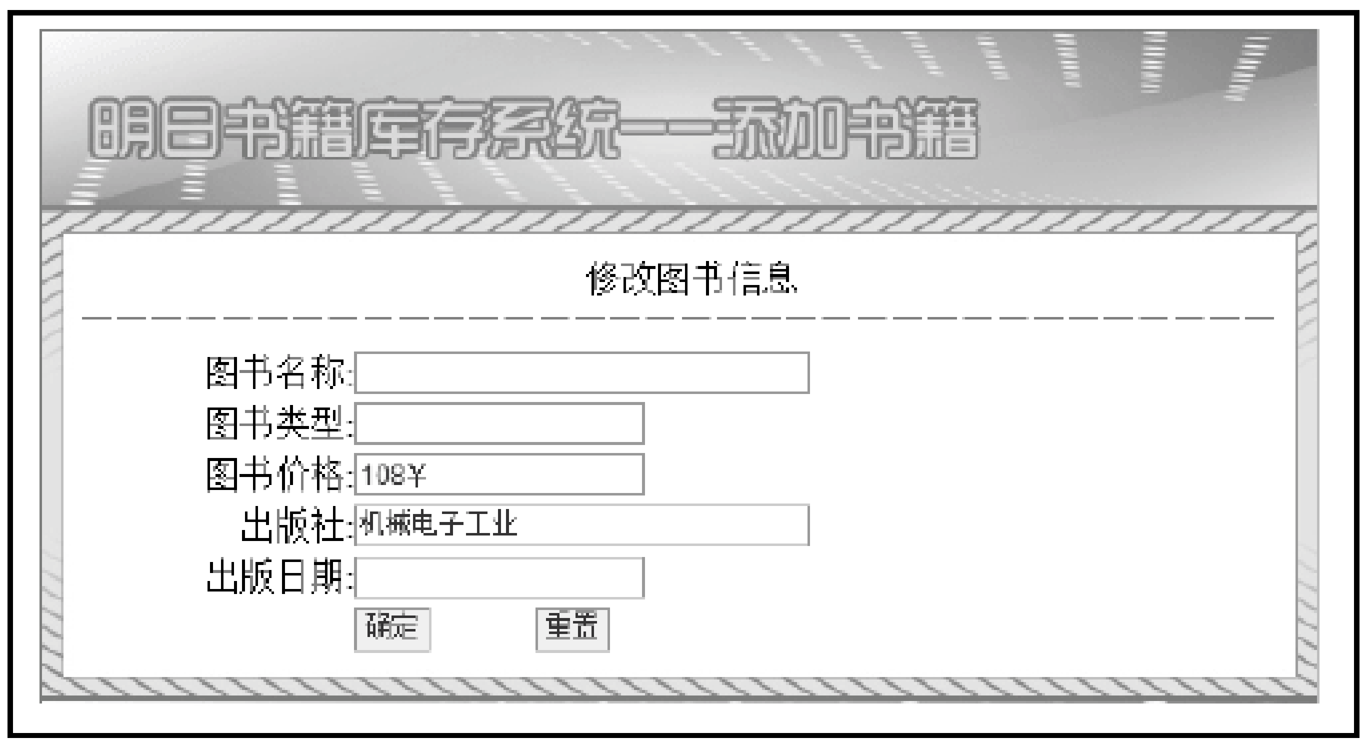


图 18.23　带记忆功能的表单

设计过程

（1）创建 index.jsp 页面，在“图书价格”和“出版社”的<input>标记中 class 属性调用 css 样式，关键代码如下：

```
<td height="22" align="left" ><input type="text" name="txt_unit"   id="txt_unit" class="saveHistory"/>
<td height="22" colspan="3" align="left"><input name="txt_co" type="text" class="saveHistory" id="txt_co" size="35">
```

（2）实现带记忆功能的表单，关键代码如下：

```
<style type="text/css">
 <meta content="history" name="save">
.saveHistory{
        behavior:url(#default#savehistory);
}
-->
</style>
```

秘笈心法

<meta>用来模拟 HTTP 协议的响应报文，是指定所提供信息的类型，应用于网页的<head>与</head>之间，属性有以下两种。

- name：常用的选项有 keywords（关键字）、description（网站内描述）、author（作者）、robots（机器人）等。
- content：根据 name 项的定义来确定此项写什么样的字符串。

实例 491　防止表单重复提交

光盘位置：光盘\MR\18\491

初级

实用指数：★★★

实例说明

用户在提交表单时，如果单击“提交”按钮后表单没有立即提交，用户则会多次单击“提交”按钮，这样就会多次提交表单内容，造成意想不到的后果。运行本实例，当用户单击“确定”按钮后，提交表单的同时该按钮将被禁用，防止表单重复提交，程序的运行结果如图 18.24 所示。

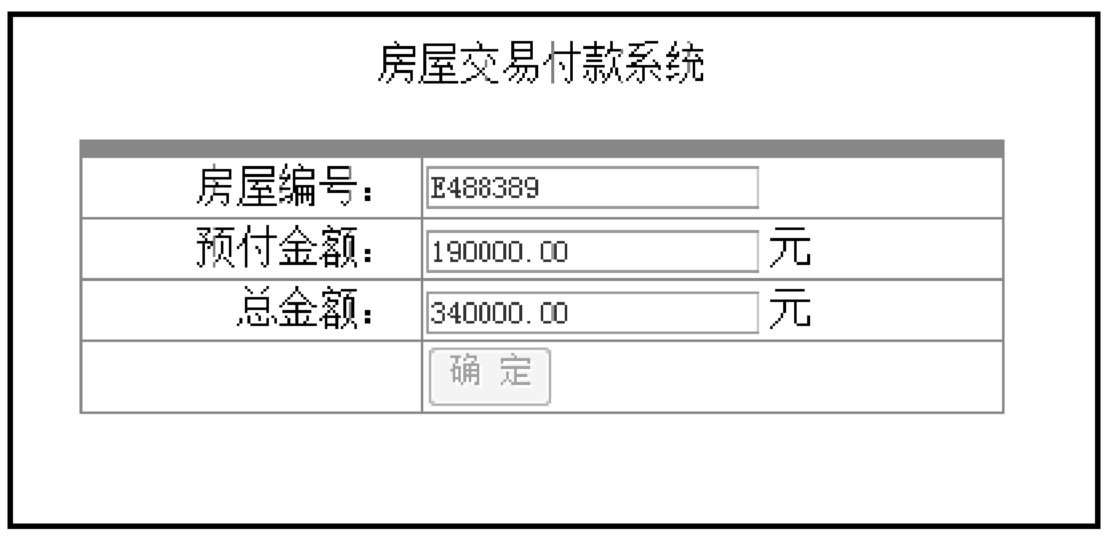


图 18.24　提交表单后禁用“确定”按钮

关键技术

本实例首先设置<input>标记的 disabled 属性值，然后再调用表单的 submit()提交方法。页面中的按钮为普通按钮，即<input>标记的类型为 button。

设计过程

创建 index.jsp 页面，在<input>添加提交表单后禁用此按钮的方法，关键代码如下：

```
onclick="this.disabled=true; this.form1.submit();"
```

秘笈心法

在 onClick 单击鼠标事件中，首先把“确定”按钮设置为可用，在提交表单后设置其 disabled 值为 true，即不可用状态。

实例 492　自动提交表单

光盘位置：光盘\MR\18\492

初级

实用指数：★★★

实例说明

在开发在线网络考试系统时，当考生开始填写表单的时间到达时限时，程序会自动提交表单以表示考试结束。运行本实例，在考生单击“开始考试”超链接时，表示考试开始；当剩余时间为 0 时，表单将自动提交，运行结果如图 18.25 所示。

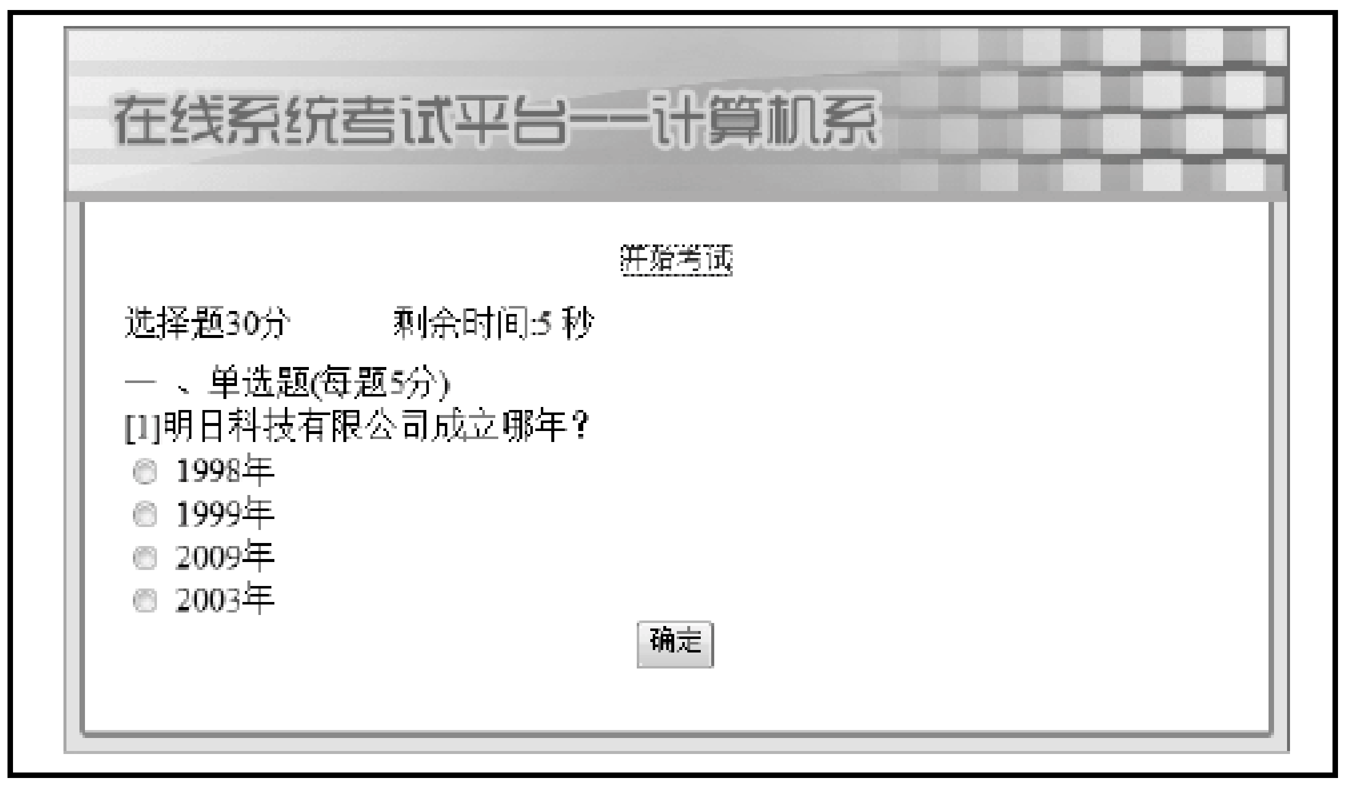


图 18.25　自动提交表单

关键技术

本实例主要是应用 JavaScript 脚本 window 对象的 setTimeout()方法。setTimeout()方法的作用是根据设定的时间间隔执行一次指定的表达式，语法如下：

```
setTimeout(表达式,延时时间[参数])
```

延时时间是以毫秒为单位的（即 1000ms=1s）。

设计过程

创建 index.jsp 页面，在“开始考试”添加超链接的 onClick 鼠标单击事件并调用自定义的 JavaScript 函数 Time_res()，关键代码如下：

```
<td height="28" align="left" valign="top">
 选择题 30 分          剩余时间:
 <font color="#FF0000" id="showtime"></font>     秒</td>
<script type="text/javascript">
var sec=10;
function Time_res(){
   document.all.show.style.display="block";
   document.all.hidden.style.display="none";
   thistime=document.form1.sec_nums.value;
   document.all.showtime.innerHTML=thistime;
   if (thistime <= 0){
      document.form1.submit();
   }
   document.form1.sec_nums.value=thistime-1;
```

```
    var timer=window.setTimeout("Time_res()",1000);
}
document.all.form1.sec_nums.value=sec;
</script>
```

秘笈心法

通过循环调用 setTimeout()方法，可以实现多次执行指定的表达式，从而判断用户的用时是否已经达到规定的时限，如果达到指定的时间，则调用表单的 submit()方法来提交表单内容。

实例 493　通过 for 循环获取表单元素的中文名称

光盘位置：光盘\MR\18\493

初级

实用指数：★★★

实例说明

当页面中包含很多表单的相关元素时，可以使用 JavaScript 脚本在 for 循环语句内获取表单元素的内容，并验证表单控件元素是否为空值。运行本实例，当用户提交表单时，如果表单中的控件元素为空字符串，则弹出提示对话框，运行结果如图 18.26 所示。



图 18.26　通过 for 循环获取表单中元素的值

关键技术

本实例主要是通过 JavaScript 的 document 对象来获取表单控件元素数组 elements[]的值，并应用 for 语句来循环元素数组的值，再应用 if 条件语句来判断各表单控件元素的值是否为空，如果为空则弹出提示对话框。

设计过程

（1）创建 index.jsp 页面文件，在页面添加表单并设置文本框的属性，关键代码如下：

```
<tr align="left" bgcolor="#FFFFFF">
    <td height="22" colspan="4" align="center">明日书籍退货系统</td>
  </tr>
  <tr align="left" bgcolor="#FFFFFF">
    <td width="183" height="22" align="left" >书籍编号:</td>
    <td width="298" height="22" ><input name="txt_snum" type="text" id="txt_snum" title="书籍编号"></td>
</tr>
  <tr align="left" bgcolor="#FFFFFF">
    <td width="183" height="22" align="left">退货数量:</td>
    <td width="298" height="22" colspan="2" ><input name="txt_nums" type="text" id="txt_nums" title="退货数量"></td>
</tr>
  <tr align="left" bgcolor="#FFFFFF">
    <td height="22" align="left">退货日期:</td>
    <td height="22" colspan="2" ><input name="txt_date" type="text" id="txt_date" title="退货日期"></td>
  </tr>
  <tr align="left" bgcolor="#FFFFFF">
    <td height="22" align="left">下次是否继续愿意购买明日图书系列:</td>
    <td height="22" colspan="2" ><input name="txt_fig" type="radio" value="1" checked>是
        <input type="radio" name="txt_fig" value="0"> 否
     </td>
  </tr>
```

（2）编写 JavaScript 脚本的 Mycheck()函数，用于实现获取表单元素的内容是否为空字符串，关键代码如下：

```
<script language="javascript">
function Mycheck(form){
        for(i=0;i<form.length;i++){
                if(form.elements[i].value==""){
                        alert(form.elements[i].title + "不能为空!");return false;}
                }
        }
}
</script>
```

秘笈心法

根据本实例的实现，读者可以开发物流信息网站中的货物详细信息页面，还可以开发游戏网站中的软件下载页面。

实例 494　可以提交到不同处理页的表单

光盘位置：光盘\MR\18\494

初级

实用指数：★★★

实例说明

在实际应用中，经常会遇到根据用户填写内容的不同，将表单提交到不同的处理页。运行本实例，根据在“提交页面”文本框中输入的页面名称，将表单提交到相应的处理页，运行结果如图 18.27 所示。

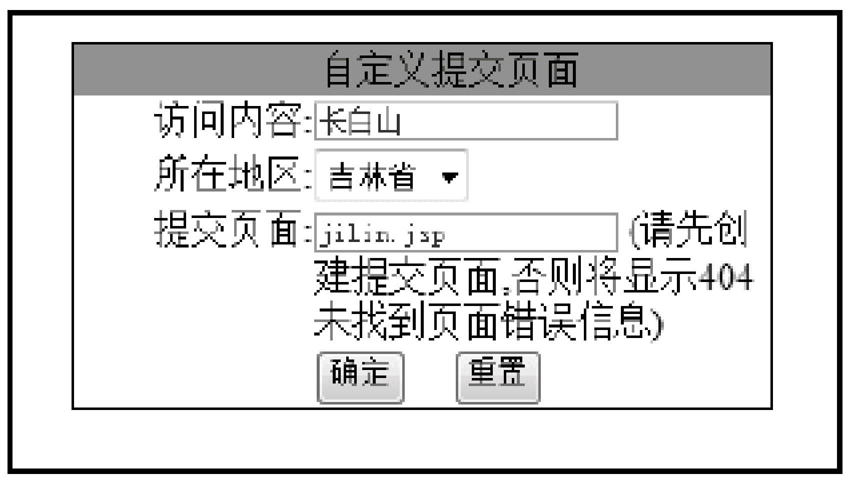


图 18.27　可以提交到不同处理页面的表单

关键技术

本实例主要通过 JavaScript 来设置表单的 action 属性，然后调用表单的 submit()提交方法实现自定义提交页面。

设计过程

首先创建 index.jsp 页面，在表单“提交页面”的<input>标记中使用 onClick 事件，并调用自定义的 Myindex()函数，关键代码如下：

```
<script type="text/javascript">
function Myindex(){
   var txt=document.form1.txt_action.value;
   document.form1.action=txt;
   document.form1.submit();
}
</script>
<input type="button" name="Submit" value="确定" onClick="Myindex();">
```

秘笈心法

根据本实例，读者可以开发根据文件类别上传文件的页面，还可以开发根据在线论坛中的用户等级来设置页面。

第 19 章

表格的操作

- 应用 JavaScript 操作表格
- 对单元格进行控制
- 表格的特殊效果

19.1　应用 JavaScript 操作表格

在开发网页时经常会开发动态网页，有些数据库的数据是常常变动的，因此为了能够方便对表格进行随意修改，就需要以指定的行数和列数动态生成或删除表格。本节将介绍如何应用 JavaScript 操作表格。

实例 495	动态制作表格 光盘位置：光盘\MR\19\495	初级 实用指数：★★★

实例说明

本实例是在页面中的行数和列数文本框中输入指定的行数和列数，并在表值文本框中输入单元格中的内容，每个单元格中的内容是以逗号“，”隔开的，单击“生成表格”按钮，将在下面生成设计好的表格。当预计行数大于实际行数时，将用空单元格补位，反之将删除多余的单元格，运行结果如图 19.1 所示。

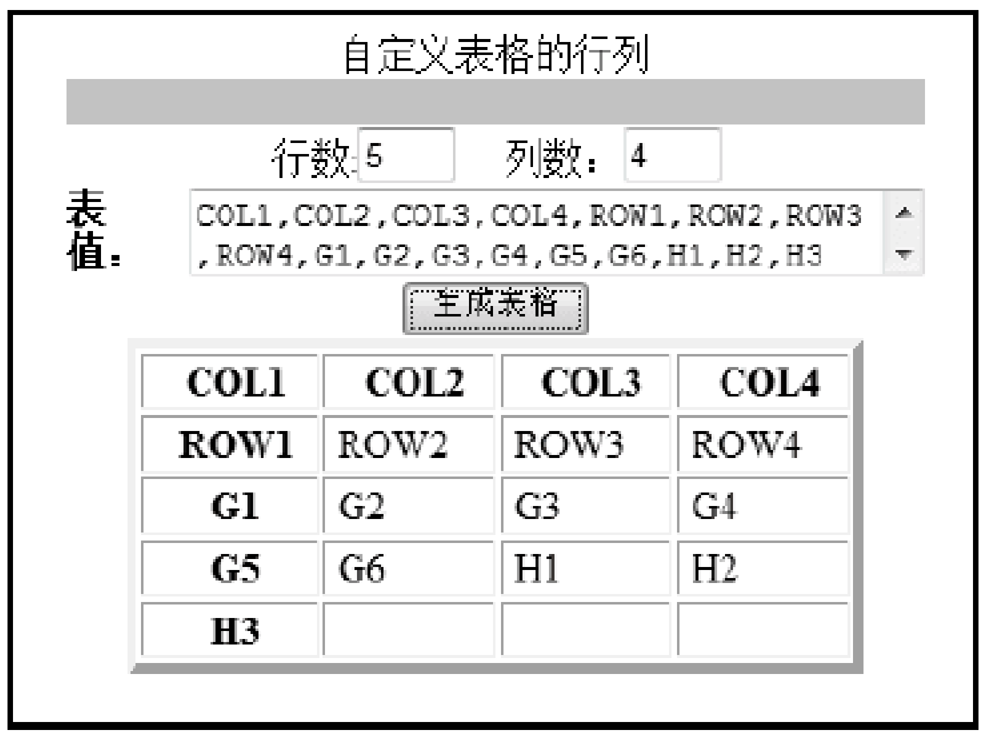


图 19.1　动态制作表格

关键技术

本实例主要是把表格相关属性<table>、<tr>、<td>标记所生成表格的代码以字符串的形式保存起来，并用层的 innerHTML 属性将表格动态生成。首先利用自定义函数 table1()将动态生成表格的首行设计为表头，然后利用自定义函数 tablevalue()生成其他行。

使用 parseInt()方法返回由字符串转换得到的整数，语法格式如下：

```
parseInt(numString,[radix])
```

参数说明

❶ numString：必选参数，为要转换数字的字符串。

❷ radix：可选参数。在 2～36 之间的表示 numString 所保存数字的进制的值。如果没有提供，则前缀为 '0x' 的字符串被当作十六进制，前缀为'0' 的字符串被当作八进制。所有其他字符串都被当作是十进制的。

设计过程

（1）创建 index.jsp 页面，在页面中添加 3 个文本框并设置相应的属性名称，关键代码如下：

```
<form name="form1" method="post" action="">
<table border="0" height="100" width="350" cellpadding="0" cellspacing="0"   align="center">
     <tr   height="20%" align="center" bgcolor=""><td colspan="3">自定义表格的行列</td></tr>
     <tr bgcolor="b0c4de" height="5%"><Td   colspan="3"> </Td></tr>
     <tr height="20%">
     <td colspan="3" align="center">行数:<input name="name1" type="text" style="width:40px" value="4">    
     列数：<input name="name2" type="text" style="width:40px" value="4"></td>
     </tr>
     <Tr height="20%">
     <td style="font: bolder">表值：</td>
     <td>
          <textarea name="name3" wrap="VIRTUAL" style="width:300px ">
               COL1,COL2,COL3,COL4,ROW1,ROW2,ROW3,ROW4,G1,G2,G3,G4,G5,G6
          </textarea>
     </td>
     </Tr>
     <tr height="15%">
     <Td colspan="3" align="center">
          <input type="button" name="Submit3" value="生成表格"
               onClick="tableclick(document.form1.name1,document.form1.name2,document.form1.name3)">
```

```
        </Td>
        </tr>
    </table>
    <div id="aa" align="center" style="font-weight: bolder"></div>
</form>
```

（2）利用 JavaScript 脚本编写用于实现动态制作表格的关键代码如下：

```
function tableclick(name1,name2,name3){//判断生成表格的条件是否正确
    Trow=name1.value;
    Tcol=name2.value;
    Tv=name3.value;
    if ((Trow=="") || (Tcol=="") || (Tv=="")){
        alert("请将制作表格的条件填写完整");
    }
    else{//将输入文本框中的行和列数转换成数字
        r=parseInt(Trow);
        c=parseInt(Tcol);
        Table1(r,c,Tv);
    }
}
```

（3）自定义 tablevalue()函数，用于生成首行以下的所有行，关键代码如下：

```
function tablevalue(a,ai,rows,col,str){
    int1=a.length;
    for (i=0;i<rows;++i){
        for (j=0;j<col;++j){
            if ((j==0)&&(ai>=int1)){
                break;
            }
            if (ai>=int1){
                str=str+"<td scope='col'> </td>";
            }
            else{
                if (j==0){
                    str=str+"<tr><th scope='col'> "+(a[ai++])+"</th>";
                }else{
                    if (j==col-1){
                        str=str+"<td scope='col'> "+(a[ai++])+"</td>";
                    }
                    else{
                        str=str+"<td scope='col'> "+(a[ai++])+"</td>";
                    }
                }
            }
        }
    }
}
```

（4）自定义 table1()函数，根据文本中的行数和列数，生成表的首行使其变为表头，关键代码如下：

```
function table1(row,col,Str1){
        var str="";
        a=new Array();
        s=new String(Str1);
        a=s.split(",");
        int1=a.length;
        ai=0;
        if(col<=int1){
            str=str+"<table width='300' border='4'>";
            for (i=0;i<col;++i){
                if (i==0){
                    str=str+"<tr><th scope='col'> "+(a[ai++])+"</th>";
                }
                else{
                    if (i==(col-1)){
                        str=str+"<th scope='col'> "+(a[ai++])+"</th></tr>";
                    }else{
                        str=str+"<th scope='col'> "+(a[ai++])+"</th>";
                    }
                }
            }
            if (int1>col){
```

```
                        if (row>1){
                                str=tablevalue(a,ai,row-1,col,str);
                        }
                }
                str=str+ "</table>";
                aa.innerHTML=str;
        }
}
```

秘笈心法

在判断生成表格的条件是否正确时，首先判断在行数、列数和表值 3 个文本框中的内容是否为空，如果为空，则用 alert()方法弹出提示框，然后再用 parseInt()方法将文本框中的行数和列数转换成数字。在表值中是以逗号“,”字符来分隔，自动生成表格中的内容使用了 split()方法，它是将一个字符串分割为子字符串，然后将结果作为字符串数组返回。

实例 496　删除表中的行

光盘位置：光盘\MR\19\496

初级

实用指数：★★★

实例说明

本实例主要实现了在表格中选中一个单元格，在单击“删除”按钮时，将删除单元格所在的行，使用该按钮可以将表格中的行全部删除，运行结果如图 19.2 所示。



图 19.2　删除表中的行

关键技术

本实例主要应用单元格的 parentElement 属性返回到表格主体，并用表格主体的 removeChild(r)方法删除所选中单元格所在的行，参数 r 表示当前单元格的父级（TR）。

设计过程

（1）创建 index.jsp 页面，在页面中添加一个表格和一个“删除”按钮并设置表格的 id 名称，关键代码如下：

```
<tr>
      <td id="txt_td13">Mr</td>
      <td id="txt_td14">Mr</td>
      <td id="txt_td15">Mr</td>
      <td id="txt_td16">Mr</td>
   </tr>
</table>
<form name="form1" method="post" action="">
        选中要删除的行：<input name="Button1" type="button" id="Button1" value="删除" onclick="removeRow()">
</form>
```

（2）利用 JavaScript 脚本编写自定义函数 select()，用于实现表中的行，关键代码如下：

```
<script language="javascript">
var lastSelection = null;
//获取选择行或单元格的参数值
function select(element) {
var e, r, c;
if (element == null){
        e = event.srcElement;
}
else {
        e = element;
}
if (e.tagName == "TD"){
        c = findCell(e);
        if (c != null) {
```

```
                if (lastSelection != null) {
                        deselectRowOrCell(lastSelection);
                }
                selectRowOrCell(c);
                lastSelection = c;
        }
}
window.event.cancelBubble = true;
}
table1.onclick=select;
```

（3）编写自定义 findCell()函数来实现查找是否为单元格，关键代码如下：

```
//查找是否有单元格
function findCell(e) {
        if (e.tagName == "TD") {
                return e;
        }
        else if (e.tagName == "BODY") {
                return null;
        }
        else{
                return findCell(e.parentElement);
        }
}
```

（4）编写自定义 selectRowOrCell()函数，用于实现改变选定行或单元格的背景颜色，关键代码如下：

```
//选择行或单元格
function selectRowOrCell(r) {
        r.runtimeStyle.backgroundColor="darkblue";
        r.runtimeStyle.color="white";
}
```

（5）编写自定义 removeRow()函数，用于实现删除单元格所在的行，关键代码如下：

```
//删除表中的行
function removeRow(){
        var r, p, nr;
        if (lastSelection==null)
                return false;
        r = lastSelection;
        if (r.tagName=="TD") {
                r = r.parentElement;
        }
        p = r.parentElement;
        alert(p.tagName);
        p.removeChild(r);
        lastSelection=null;
        return r;
}
</script>
```

秘笈心法

本实例中用到的 parsentElement 属性就是父对象，如 td 的 parsentElement 就是 tr，tr 的 parsentElement 就是 tbody，而这种方式只针对 IE。如果要在其他浏览器上使用就要使用 parsentNode，它可以兼容多个浏览器并且可以代替 parentElement 的所有功能。

实例 497	动态生成行或列 光盘位置：光盘\MR\19\497	初级 实用指数：★★★

实例说明

本实例对表格中的每一行每一列都使用“插入行”和“插入列”按钮来完成。当插入多行以后，再插入列时，每一列中的行数与插入后的行数相同，反之也一样，程序的运行结果如图 19.3 所示。

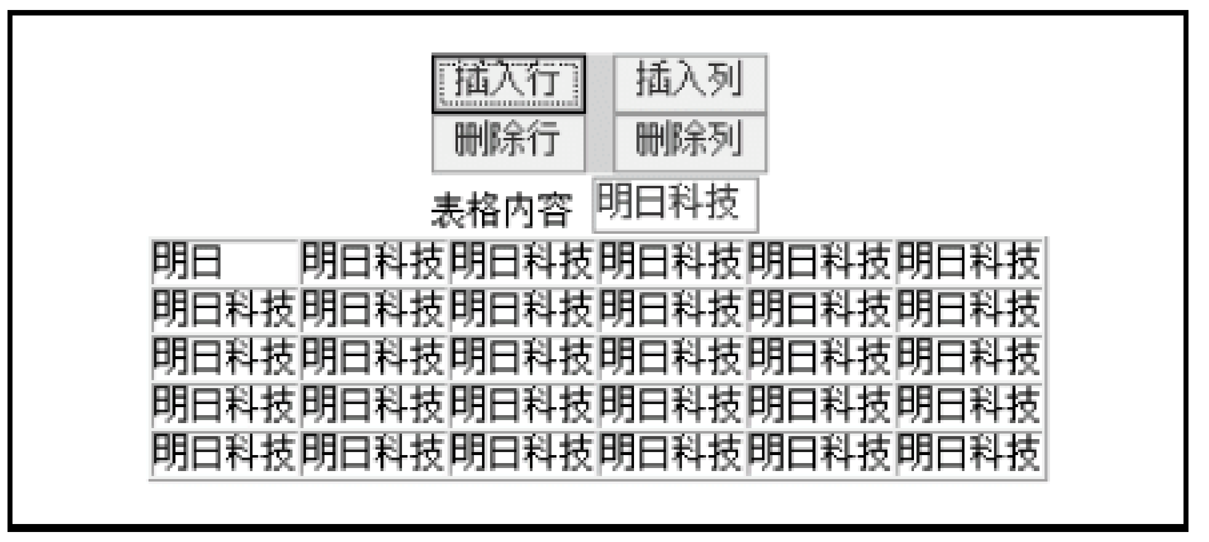


图 19.3　动态添加行或列

■ 关键技术

本实例应用了表格的 insertRow(n)方法，此方法是在表格的尾部添加行，参数 n 表示当前表格的行数。在为表格添加列时，实际上是以表格的行数为最大值，用表格的 rows[i]的 insertCell(n)方法向每行的尾部添加单元格，参数 i 表示行数，n 表示当前表格的最大列数，用表格的 row[i]的 cells[n]的 innerText 属性来动态为各单元格中添加指定的文本，i 表示指定的行，n 表示指定行中的第几个单元格。

insertRow()方法的语法如下：

```
tableobject.insertRow(index)
```

返回值：是一个 tableRow，表示插入的行。

■ 设计过程

（1）创建 index.jsp 页面，在页面中添加 4 个相应按钮并设置其相应属性，关键代码如下：

```
<table cellpadding="0" cellspacing="0" border="0" align="center"   bgcolor="d3d3d3">
<Tr valign="bottom" ><td>
<input type="button" value="插入行" onclick="insertrow()">  
<input type="button" value="插入列" onclick="insertcell()">
</td></Tr>
<Tr><td>
<input type="button" value="删除行" onclick="deleterow()">  
<input type="button" value="删除列" onclick="deletecell()">
</td></Tr>
<tr bgcolor="#FFFFFF">
<td>表格内容
<input name="text1" width="40px" size="5" value="Mr_pro" > </td></tr>
</table >
<table id="tt" cellpadding=0 cellspacing=0 border="1" align="center">
<tr><td>明日</td></tr>
</table>
```

（2）利用 JavaScript 脚本，编写实现手动插入或删除表格行或列，自定义插入行 insertrow()函数，应用于在已有单元格的后面插入行，关键代码如下：

```
<script language="JavaScript">
function insertrow(){
 text1.value==""?str=" ":str=text1.value;
 var n=tt.rows.length;                    //行长度
 tt.insertRow(n);                         //插入一行
 for (i=0;i<=tt.rows(0).cells.length-1;i++){
        tt.rows[n].insertCell(i);         //在插入的行中插入单元格
        tt.rows[n].cells[i].innerText=str;  //放入 Text
 } }
```

（3）自定义插入列 insertcell()函数，应用于在已有单元格的后面插入列，关键代码如下：

```
function insertcell(){
 text1.value==""?str=" ":str=text1.value;
 var n=tt.rows[0].cells.length;           //0 行的单元格长度
 for (i=0;i<=tt.rows.length-1;i++){
        tt.rows[i].insertCell(n);         //在某行中插入单元格
        tt.rows[i].cells[n].innerText=str;  //放入 Text
 }
}
```

（4）自定义删除行 deleterow()函数，应用于从表的下边开始向上删除行，自定义删除列 deletecell()函数，

从表的右边开始向左边删除列，关键代码如下：

```
function deleterow(){
 var n=tt.rows.length;                //行长度
 if(n==1) return;
 tt.deleteRow(n-1);                   //删除某一行
}
function deletecell(){
 var n=tt.rows[0].cells.length;       //0 行的单元格长度
 if(n==1) return;
 for (i=0;i<=tt.rows.length-1;i++){
        tt.rows[i].deleteCell(n-1);   //删除某行的某一单元格
 }
}
</script>
```

秘笈心法

本实例中在单元格插入行和列时，都用到一个 innerText 方法，它是从起始位置到终止位置的内容（只对于 IE 浏览器），如果要在其他浏览器上使用就要用到 innerHTML，innerHTML 是符合 W3C 的标准属性。

实例 498 合并单元格

光盘位置：光盘\MR\19\498

初级

实用指数：★★★

实例说明

本实例是把指定列中连续相同的单元格进行合并，这样可以使表格更加清晰明了，方便用户浏览阅读，程序的运行结果如图 19.4 所示。

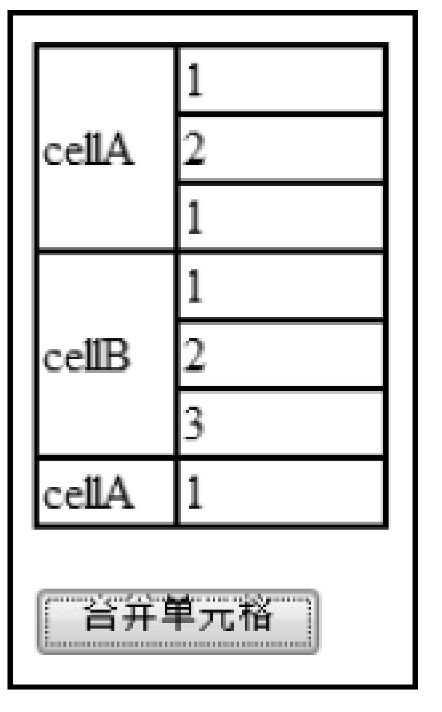


图 19.4　合并单元格效果

关键技术

本实例主要使用表格中的 rows 的 length 属性来获得表格的总行数，在用表格中的 rows[nrow]数组的 cell[ncol]数组的 innerText 属性来获取表格中行的指定单元格的内容，并判断纵向单元格是否相同，如果相同，则使用表格的 row[i]的 deleteCell(ncol)方法，将连续相同的列中的单元格删除。

设计过程

（1）创建 index.jsp 页面，在页面中添加一个表格和按钮，关键代码如下：

```
<table id="table1" width="14%" cellpadding="1"   bordercolor="#000000" bgcolor="#000000">
        <tr bgcolor="#FFFFFF">
              <td     width="40%">cellA</td>
              <td     width="60%">1</td>
        </tr>
        <tr bgcolor="#FFFFFF">
              <td     width="40%">cellA</td>
              <td     width="60%">2</td>
        </tr>
<input type="button" value="合并单元格" onclick="merge()">
```

（2）使用 JavaScript 编写实现合并单元格，关键代码如下：

```
<input type="button" value="合并单元格" onclick="merge()">
<script language="javascript">
function merge(){
 var rows=table1.rows;
 var nrow=0,nlastrow;
 var ncol=0;
 while(nrow<rows.length){
        nlastrow=nrow++;
        while(nrow<rows.length && rows[nlastrow].cells[ncol].innerText==rows[nrow].cells[ncol].innerText)
               nrow++;
```

```
        if(nrow-nlastrow>1){
                for(var i=nlastrow+1;i<nrow;i++){rows[i].deleteCell(ncol);}
                rows[nlastrow].cells[ncol].rowSpan=nrow-nlastrow;
        }
}
}
</script>
```

秘笈心法

在实现合并单元格时，应用了 while()语句来具体判断将后一个单元格添加到要合并的单元格中，然后再使用 for()循环语句把添加后的单元格使用自定义 JavaScript 脚本函数 deleteCell()删除。

实例 499　在表格中添加行及单元格

光盘位置：光盘\MR\19\499

初级

实用指数：★★★

实例说明

在表格中添加行，实际上就是在表格的下面动态添加单元格，并在刚添加的单元格处添加信息，程序的运行结果如图 19.5 所示。

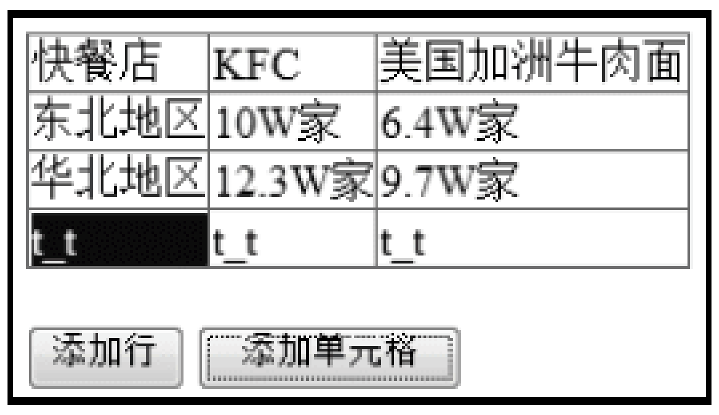


图 19.5　在表格中添加行及单元格

关键技术

本实例主要使用 document 对象的 createElement("tr")方法创建表格的行的节点，再应用 document 对象的 createElement("td")方法在创建行中创建一个单元格节点，并用<td>标记的 insertBefore(text,null)方法向已创建的单元格中添加信息，用<tr>标记的 insertBefore(nc,c)方法添加单元格。

设计过程

（1）创建 index.jsp 页面，在页面中添加表格及两个按钮，并设置表格的相关属性，关键代码如下：

```
 <tr>
     <td id="cell7">华北地区</td>
     <td id="cell8">12.3W 家</td>
     <td id="cell9">9.7W 家</td>
  </tr>
</table>
<form name="form1" method="post" action="">
  <input name="Button1" type="button" id="Button1" value="添加行" onclick="addRow()">
  <input name="Button2" type="button" id="Button2" value="添加单元格" onclick="addCell()">
</form>
```

（2）使用 JavaScript 脚本编写对表格添加行和单元格，使用自定义函数 addRow()为表格添加行，关键代码如下：

```
//为表格添加行
function addRow() {
 var r, p, nr;
 if (lastSelection==null) {r=null;p=table1.children[0];}
 else {
        r = lastSelection;
        if (r.tagName=="TD") {r=r.parentElement;}
        p = r.parentElement;
 }
 nr = document.createElement("TR");
 p.insertBefore(nr, r);
 select(nr);
 addCell();
 return nr;
}
```

（3）自定义函数 addCell()，用于在新添加行的位置插入一个单元格，关键代码如下：

```
//在添加的行中插入一个单元格
function addCell(){
var r, p, c, nc, text;
if (lastSelection == null)    return false;
r = lastSelection;
if (r.tagName == "TD") {r = r.parentElement;c = lastSelection;}
else {c = null;}
nc = document.createElement("TD");
text = document.createTextNode("t_t");
nc.insertBefore(text, null);
r.insertBefore(nc, c);
select(nc);
return nc;
}
```

本实例用到的其他自定义函数在上述范例中都有详细说明，为了节省空间这里不再重复介绍。

秘笈心法

在实现本实例时，在 JavaScript 的函数中用到了 tagName 属性以及 parentElement 属性。tagName 属性可以返回当前对象的类型字符串，如当前对象为<TD>标签，那么调用 tagName 返回值就是 TD；parentElement 属性用于返回当前对象表示的标签的父标签，如当前对象为<TD>标签，那么调用 parentElement 属性返回的就是表示<TR>标签的对象。

实例 500	删除表中的单元格 光盘位置：光盘\MR\19\500	初级 实用指数：★★★

实例说明

本实例是用鼠标选中表中的单元格，然后单击“删除”按钮，将删除表中的单元格，后面的单元格将向前移位，移到刚刚删除的单元格位置，运行结果如图 19.6 所示。



图 19.6　删除表中的单元格

关键技术

本实例主要使用<td>标记的 parentElement 属性来获得父级标记<tr>，再使用<tr>标记的 removeChild(c)方法将选中的单元格删除。

设计过程

（1）创建 index.jsp 页面文件，在页面添加表格及按钮，关键代码如下：

```
<tr>
      <td id="cell7">单元格 7</td>
      <td id="cell8">单元格 8</td>
      <td id="cell9">单元格 9</td>
   </tr>
</table>
<form name="form1" method="post" action="">
   <input name="Button1" type="button" id="Button1" value="删除" onclick="removeCell()">
</form>
```

（2）使用 JavaScript 脚本编写删除表中的单元格，自定义函数 removeCell()，关键代码如下：

```
//删除表中的单元格
function removeCell(){
var c, p, nr;
if (lastSelection == null)    return false;
c = lastSelection;
if (c.tagName != "TD") { return null; }
p = c.parentElement;
```

```
p.removeChild(c);
lastSelection = null;
return c;
}
</script>
```

秘笈心法

在核心 JavaScript 中有一个操作 DOM 节点的方法就是 removeChild()，简单地说是实现移除指定的子节点，我们可以先去找到要删除节点的父节点，然后在父节点中运用 removeChild()方法移除子节点。

实例 501　从表格最下面向上删除单元格

光盘位置：光盘\MR\19\501

初级

实用指数：★★★

实例说明

本实例实现了在表格中以由下向上的顺序依次删除单元格，当单击“拆除”按钮时，表格的最底层一行将全部被删除，再次重复单击“拆除”按钮，这时原来倒数第二行的单元格变为最后一层，则被删除，程序的运行结果如图 19.7 和图 19.8 所示。

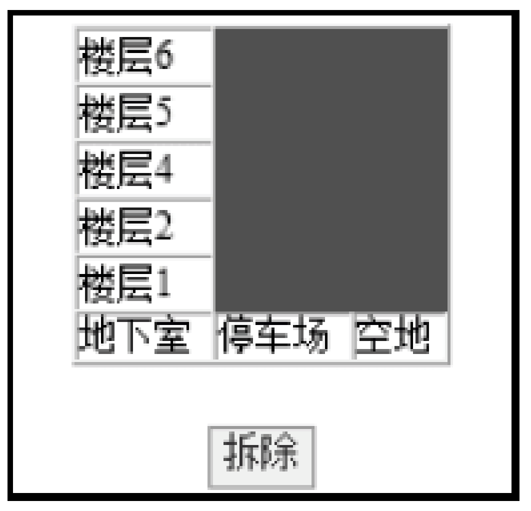


图 19.7　单击“拆除”按钮之前

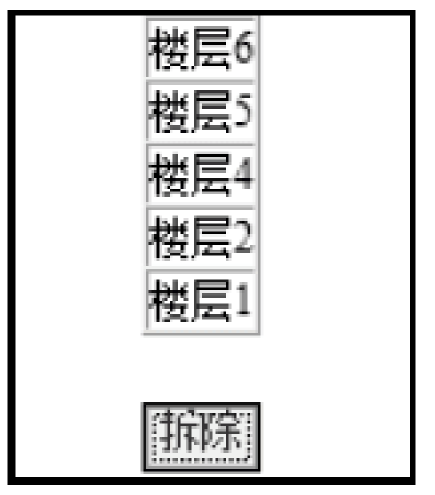


图 19.8　单击“拆除”按钮之后

关键技术

本实例主要使用 JavaScript 脚本语言，在自定义函数 deleterow()中自定义了一个变量 n，并用行长度值赋值给此变量，获取行长度的方法如下：

```
tt.rows.length
```

其中，tt 为表格的 id 名称，rows.length 为表格的行长度。

设计过程

（1）创建 index.jsp 页面文件，在页面中添加表格并设置表格的 id 名称及其属性，关键代码如下：

```
<table id="tt" cellpadding="0" cellspacing="0" border="1" align="center" bgcolor="#8b4513">
        <tr bgcolor="#ffffff" >
            <Td>楼层 6</Td>
        </tr>
        <tr bgcolor="#ffffff">
            <Td>楼层 5</Td>
        </tr>
        <tr bgcolor="#ffffff">
            <Td>楼层 4</Td>
        </tr>
        <tr bgcolor="#ffffff">
            <Td>楼层 2</Td>
        </tr>
        <tr bgcolor="#ffffff">
            <Td>楼层 1</Td>
        </tr>
        <tr bgcolor="#ffffff" >
            <Td>地下室</Td>
            <Td>停车场</Td>
```

```
                    <Td>空地</Td>
            </tr>
</table>
    <p align="center"><input type="button" value="拆除" onclick="deleterow()" /></p>
```

（2）使用 JavaScript 脚本编写实现由下向上依次删除表格中的单元格效果，自定义函数 deleterow()，关键代码如下：

```
<script type="text/javascript">
        function deleterow(){
                var n = tt.rows.length;;
                if(n==1)return;
                tt.deleteRow(n-1);
        }
 </script>
```

秘笈心法

本实例中，在“拆除”按钮的<input>标记中使用了 onClick 鼠标单击事件，当单击“拆除”按钮后，调用了自定义函数 deleterow()方法。所谓 onClick 事件，就是在用户单击鼠标左键时发生（如果单击鼠标右键则不会发生），当用户的焦点在按钮上，并按了 Enter 键，同样会触发这个事件。

实例 502　在表格的右侧动态添加列

光盘位置：光盘\MR\19\502

初级

实用指数：★★★

实例说明

本实例将介绍如何在原有表格基础上，实现在表格的右侧动态添加列。运行本实例，单击“添加”按钮时，则会在表格单元格所在行尾自动插入一个单元格，程序的运行结果如图 19.9 所示。



图 19.9　在表格的右侧添加列

关键技术

本实例主要使用 JavaScript 脚本，首先获取表格行的单元格长度，再使用 for 语句来循环行的最后一个元素，此时再执行 insertCell(n)方法。

获取表格的行中单元格个数的语法格式如下：

```
tt.rows[0].cells.length
```

设计过程

（1）创建 index.jsp 页面文件，在此页面中添加表格并设置 id 名称及相关属性，关键代码如下：

```
        <Td><input type="button" value="添加" onclick="insertcell()" /></Td>
        <td><input value="单元格" name="text1" size="5" /></td>
        </Tr></table>
<table id="tt" cellpadding="0" cellspacing="0" border="1">
        <Tr>
                <td>明日</td>
        </Tr>
        <Tr>
                <td>科技</td>
        </Tr>
</table>
```

（2）使用 JavaScript 脚本编写实现在表格内的单元格插入列的函数 insertCell()，用于在单元格的后面插入列，关键代码如下：

```
<script type="text/javascript">
function insertcell(){
 text1.value==""?str=" ":str=text1.value;
 var n=tt.rows[0].cells.length;              //0 行的单元格长度
 for (i=0;i<=tt.rows.length-1;i++){
```

```
        tt.rows[i].insertCell(n);              //在某行中插入单元格
        tt.rows[i].cells[n].innerText=str;    //放入 Text
}
}
</script>
```

秘笈心法

在 JavaScript 中，innerText 属性用于动态修改 HTML 标签的内容，应用它可以使代码更加简洁。另外，使用 innerText 会自动将小于号、大于号、引号和&符号写进 HTML 编码。

实例 503　从表格的右侧依次删除所有列

光盘位置：光盘\MR\19\503

初级

实用指数：★★★

实例说明

在页面显示大量数据时，对于已查出并显示在页面上的信息来说，删除一列或所有列的功能会十分有用。运行本实例，当单击“删除列”按钮时，表格的最后一列将被删除，程序的运行结果如图 19.10 和图 19.11 所示。

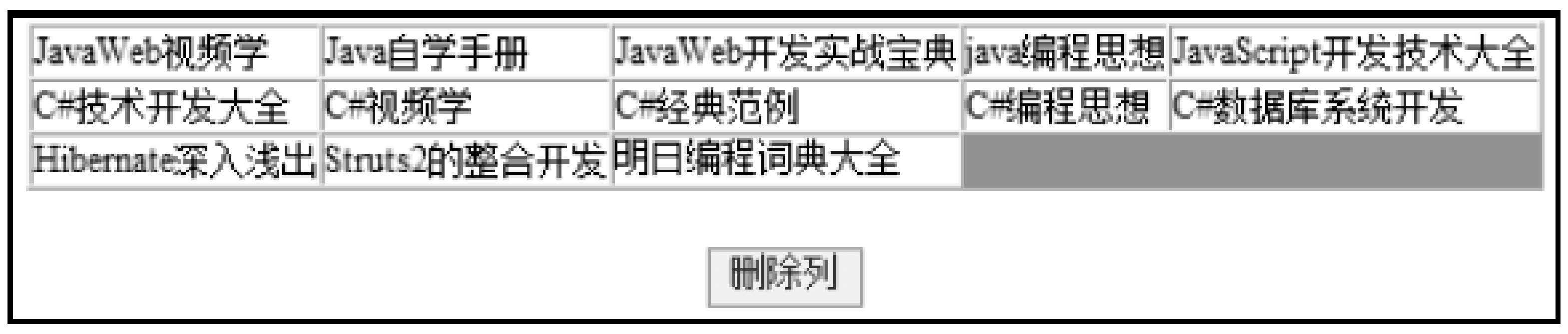


图 19.10　删除表格的列之前

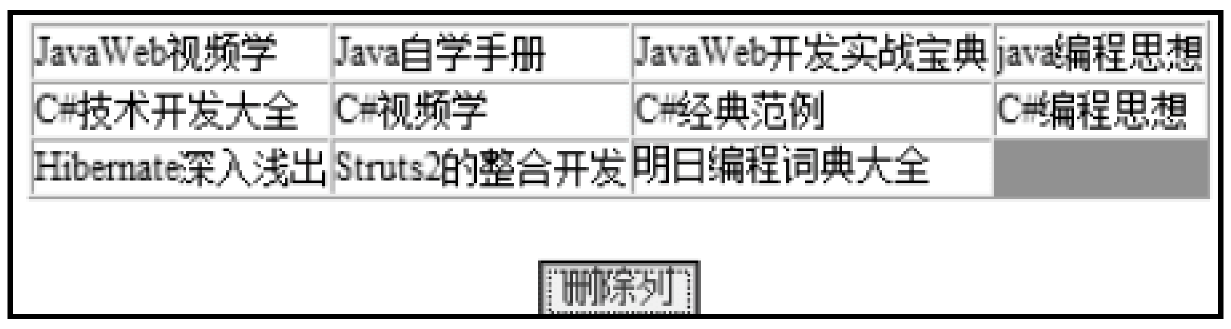


图 19.11　单击“删除列”按钮之后

关键技术

本实例主要应用 JavaScript 自定义函数 deletecell()，首先定义一个变量 n，然后把行的单元格的长度赋值给此变量，使用 if 条件判断语句，如果变量 n 等于 1，则返回。最后使用 for 语句循环输出 row[i]中所有行单元格的长度，再调用此自定义函数 deletecell()减 1，每执行一次便删除行尾处单元格的列。

设计过程

（1）创建 index.jsp 页面文件，在页面添加一个表格并设置 id 名称及其相关属性，关键代码如下：

```
<table border="1" cellpadding="0" cellspacing="0" id="tt" align="center" bgcolor="#6495ed">
        <tr bgcolor="#FFFFFF">
                <td>JavaWeb 视频学</td>
                <td>Java 自学手册</td>
                <td>JavaWeb 开发实战宝典</td>
                <td>java 编程思想</td>
                <td>JavaScript 开发技术大全</td>
        </tr>
                <tr bgcolor="#FFFFFF">
                <td>C#技术开发大全</td>
                <td>C#视频学</td>
                <td>C#经典范例</td>
                <td>C#编程思想</td>
                <td>C#数据库系统开发</td>
        </tr>
                <tr bgcolor="#FFFFFF">
                <td>Hibernate 深入浅出</td>
                <td>Struts2 的整合开发</td>
                <td>明日编程词典大全</td>
        </tr>
    </table>
<p align="center"><input type="button" value="删除列" onclick="deletecell()" /></p>
```

（2）使用 JavaScript 脚本实现删除表格一列的效果，并在<input>标记中调用此函数，关键代码如下：

```
<script type="text/javascript"   >
   function deletecell(){
 var n=tt.rows[0].cells.length;       //0 行的单元格长度
 if(n==1) return;
 for (i=0;i<=tt.rows.length-1;i++){
         tt.rows[i].deleteCell(n-1);   //删除某行的某一单元格
 }
 }
</script>
```

秘笈心法

本实例使用了 row[0].cells.length，它表示表格的行的单元格个数，再进行循环条件 tt.rows.length−1，就是要把 0 行删除，找到第一个单元格，length 默认是 0 到最大值。

实例 504　在表格中动态添加行　初级

光盘位置：光盘\MR\19\504　实用指数：★★★

实例说明

在开发网站时，要求表格具有灵活性，使用户可随意添加表格行是十分常见的。运行本实例，当加载完页面时，单击“添加”按钮，在已有“明日科技”单元格下面将自动插入一行单元格，程序的运行结果如图 19.12 和图 19.13 所示。



图 19.12　未插入行之前



图 19.13　单击“添加”按钮之后

关键技术

本实例主要应用 JavaScript 脚本，首先获取行的长度并赋值给自定义变量 n，然后调用 tt.insertRow(n)插入一行，使用 for 语句循环输出所有单元格的行，并在要插入行的位置插入单元格，最后把 rows[n]数组放在 innerText 中。

设计过程

（1）创建 index.jsp 页面文件，在页面中添加表格及其 id 名称属性，关键代码如下：

```
<div><input type="button" value="添加" onclick="insertrow()"   /></div>
              <input type="text"   id="text1" value="单元行" size="5"/>
        <table border="1" cellpadding="0" cellspacing="0" id="tt">
              <Tr>
                     <Td>明日科技</Td>
              </Tr>
</table>
```

（2）使用 JavaScript 脚本，自定义函数 insertrow()实现在已有单元格的下面插入行，关键代码如下：

```
<script language="JavaScript">
function insertrow(){
 text1.value==""?str=" ":str=text1.value;
 var n=tt.rows.length;                        //行长度
 tt.insertRow(n);                             //插入一行
 for (i=0;i<=tt.rows(0).cells.length-1;i++){
         tt.rows[n].insertCell(i);            //在插入的行中插入单元格
         tt.rows[n].cells[i].innerText=str;   //放入 Text
 }
 }
</script>
```

秘笈心法

在已有单元格下面插入行主要是把单元格先保存在一个数组当中，封装以后对其内的所有元素进行循环。

19.2　对单元格进行控制

实例 505　选定表格中的单元格　初级

光盘位置：光盘\MR\19\505　实用指数：★★★

实例说明

在页面上浏览表格中的数据时，有时会忘记看到了哪条数据，这时就应该用鼠标选中已读取到的数据并改变表格中单元格的背景颜色，以提示浏览者数据的位置，方便以后继续读取，程序的运行结果如图 19.14 所示。

年份	产品名称	是否出口	总计销量
2000年	张裕葡萄酒	己出口	25W瓶
2002年	通化葡萄酒	未出口	12W瓶

图 19.14　已被单击单元格

关键技术

本实例主要应用了在表格的 onClick 单击事件的 event 对象中的 srcElement 属性来获取发生事件的文档元素，并将其保存在自定义变量 e 中，使用 e.tagName()来判断发生事件的文档元素是否在表格的单元格上，并用变量 e 中的 runtimeStyle 样式中的 color 和 backgroundColor 属性来改变当前选中单元格的前景色和背景色。window 对象的 lastSelection 属性用于获取最后一次选中的单元格焦点。

设计过程

（1）创建 index.jsp 页面，在页面中添加一个表格，关键代码如下：

```
<table id="table1" border="0" cellpadding="1" cellspacing="1"
bordercolor="#000000" bgcolor="#000000" align="center">
  <tr bgcolor="#FFFFFF">
    <td>年份</td>
    <td>产品名称</td>
    <td>是否出口</td>
    <Td>总计销量</Td>
  </tr>
  <tr bgcolor="#FFFFFF">
    <td>2000 年</td>
    <td>张裕葡萄酒</td>
    <td>己出口</td>
    <td>25W 瓶</td>
  </tr>
  <tr bgcolor="#FFFFFF">
    <td>2002 年</td>
    <td>通化葡萄酒</td>
    <td>未出口</td>
    <td>12W 瓶</td>
  </tr>
</table>
```

（2）利用 JavaScript 脚本编写实现选中单元格用自定义函数来保存发生事件的文档元素信息，关键代码如下：

```
<script language="javascript">
var lastSelection = null;
function select(element) {
var e, r, c;
if (element == null){e = event.srcElement;}//获取 body 元素的原始记录
else { e = element; }
if (e.tagName == "TD"){
```

```
        c = findCell(e);
        if (c != null) {
                if (lastSelection != null) {deselectRowOrCell(window.lastSelection);}
                selectRowOrCell(c);
                lastSelection = c;
        }
}
window.event.cancelBubble = true;       //取消冒泡语句，用于防止向下一个外层对象冒泡
}
table1.onclick=select;
function findCell(e) {
if (e.tagName == "TD") {
        return e;
}
else if (e.tagName == "BODY") {
        return null;
}
else{
        return findCell(e.parentElement);
}
}
function selectRowOrCell(r) {
r.runtimeStyle.backgroundColor="darkblue";
r.runtimeStyle.color="white";
}
function deselectRowOrCell(r) {
r.runtimeStyle.backgroundColor = "";
r.runtimeStyle.color = "";
}</script>
```

秘笈心法

本实例应用了冒泡语句来防止向下一个外层对象冒泡，如果事件是起泡类型，想停止起泡只有调用 stop Propagation()方法才能停止。

实例 506	可左右移动单元格的信息 光盘位置：光盘\MR\19\506	初级 实用指数：★★★

实例说明

本实例主要在单元格获得焦点后，通过页面中的两个按钮来使单元格的信息可以左右移动，在信息移动到最左边或最右边时则无法再移动，程序的运行结果如图 19.15 和图 19.16 所示。



图 19.15　单元格被选中时

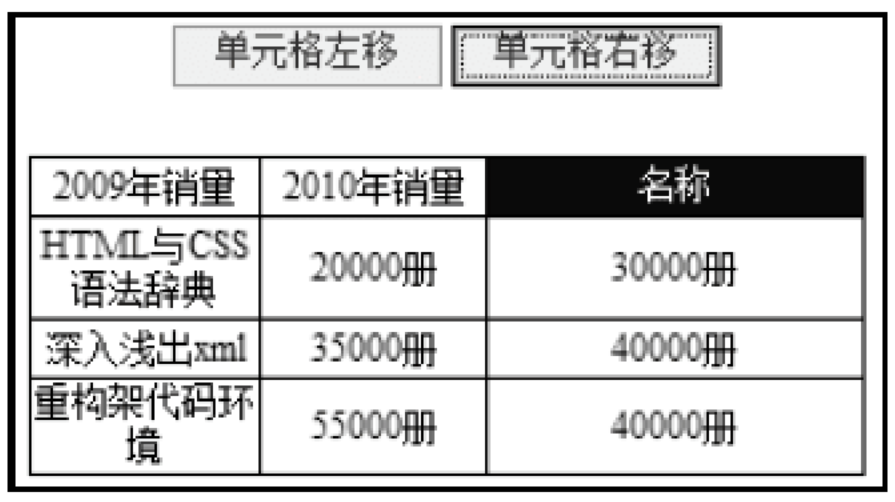


图 19.16　移动到最右边的效果

关键技术

本实例主要是利用 event 对象的 srcElement 属性获取发生事件的文档元素，并将其数据内容保存在变量 c 中，在移动单元格信息之前，先使用 c.tagName 来判断当前焦点的标签名是否为 td（单元格的标记），当为 td 时，说明表格中的单元格获得了焦点，再使用 c.previousSibling 或 c.nextSibling 来获取当前单元格同一行的前一个或下一个单元格的位置，并将其保存在 ls 变量中，再用 ls.parentElement 来返回当前位置的上一级（即 tr）。

设计过程

（1）创建 index.jsp 页面，在页面添加一个表格和两个按钮，并在自定义函数中调用 moveLeft()和 moveRight()，关键代码如下：

```
<form name="form1" method="post" action="">
<table align="center"><tr><td>
  <input name="Button1" type="button" id="Button1" value="单元格左移" onclick="moveLeft()">
  <input name="Button2" type="button" id="Button2" value="单元格右移" onclick="moveRight()
  ">
  </td></tr></table>
</form>
<tr bgcolor="#FFFFFF" align="center" >
     <td>HTML 与 CSS 语法辞典</td>
     <td>20000 册</td>
     <td>30000 册</td>
  </tr>
  <tr bgcolor="#FFFFFF" align="center" >
     <td>深入浅出 xml</td>
     <td>35000 册</td>
     <td>40000 册</td>
  </tr>
  <tr bgcolor="#FFFFFF" align="center" >
     <td>重构架代码环境</td>
     <td>55000 册</td>
     <td>40000 册</td>
  </tr>
```

（2）编写 JavaScript 脚本自定义 select()函数用来保存发生事件的文档元素信息，关键代码如下：

```
<script language="javascript">
var lastSelection = null;
//获取选择行或单元格的参数值
function select(element) {
 var e, r, c;
 if (element == null){e = window.event.srcElement;}
 else {    e = element;    }
 if (e.tagName == "TD") {
         c = findCell(e);
         if (c != null) {
                 if (lastSelection != null) {deselectRowOrCell(lastSelection);}
                 selectRowOrCell(c);
                 lastSelection = c;
         }
 }
 window.event.cancelBubble = true;}
table1.onclick=select;
```

（3）自定义 moveLeft()函数，用于将当前单元格的内容向左移动，关键代码如下：

```
//单元格向左移
function moveLeft() {
 var c, p, ls;
 if (lastSelection == null)
         return false;
 c = lastSelection;
 if (c.tagName != "TD") {
           return null;
 }
 ls = c.previousSibling;
 if (ls == null)
           return null;
 p = ls.parentElement;
 p.insertBefore(c, ls);
 return c;
}
```

（4）自定义 moveRight()函数，用于将当前单元格的信息向右移动，关键代码如下：

```
//单元格向右移
function moveRight() {
```

```
var c, p, ls;
if (lastSelection == null)
        return false;
c = lastSelection;
if (c.tagName != "TD") {
        return null;
}
ls = c.nextSibling;
if (ls == null)
        return null;
p = ls.parentElement;
ls = ls.nextSibling;
p.insertBefore(c, ls);
        return c;
}
</script>
```

秘笈心法

应用 insetBefore()方法可以将当前节点插入到指定节点之前，使单元格的内容向左或向右移动。Previous Sibling 是指获取前一个对象，而 nextSibling 是指获得下一个对象。

实例 507 使用键盘使单元格焦点随意移动

光盘位置：光盘\MR\19\507

初级

实用指数：★★★

实例说明

本实例将介绍如何实现使用键盘使单元格焦点随意移动。运行本实例，使用键盘的方向键来获取焦点进行上下左右的移动，当持续按方向键时，单元格焦点将在表格中进行循环移动，程序的运行结果如图 19.17 所示。



	音乐频道	体育频道
14: 00--16: 00	最新流行歌曲	南非世界杯回放
16: 00--18: 00	最新演唱会精选	中国男篮精英赛

图 19.17　使用键盘使单元格焦点随意移动

关键技术

本实例主要使用 JavaScript 脚本创建一个 ObjTable 对象，用这个对象来对表格中的焦点移动进行控制的实现过程如下：

（1）通过当前表格的 rows 的 length 属性来获取表格的所有行数，通过某一行的单元格个数来获取表格的总列数方法为（sTable.rows[0].cells.length）。

（2）创建一个全局 ObjTable 对象，并将表格的行数和列数分别赋给 ObjTable 对象的 rowCount 和 colCount 属性。

（3）通过 ObjTable 的单击事件（onClick）来获得表格中的焦点。

（4）通过 ObjTable 的键盘的方向键向下事件（onKeyDown），对键盘的操作进行监控，其中“→”键的键值为 3，“←”键的键值为 2，“↓”键的键值为“1”，“↑”键的键值为 0，Enter 键的键值为 13，通过键盘上移动键对表格中单元格的焦点进行移动，用 Enter 键来实现换行的操作。

（5）通过对键盘的操作，对当前单元格坐标进行修改，并用 ObjTable 对象的 getNode()方法重新获取单元格的焦点。

设计过程

（1）创建 index.jsp 页面，在页面中添加一个表格，并设置其 id 属性为 table1，关键代码如下：

```
<table id="table1" width="360" border="0" cellpadding="1" cellspacing="1" align="center"
 bordercolor="#000000" bgcolor="#000000">
  <tr bgcolor="#FFFFFF">
    <td width="120"></td>
```

```
    <td width="72">音乐频道</td>
    <td width="71">体育频道</td>
  </tr>
```

（2）利用 JavaScript 脚本编写通过键盘方向键使单元格焦点进行移动，自定义一个 ObjTable 对象，用于存放页面中表格的相关信息，关键代码如下：

```
<script language="JavaScript">
var ObjTable = new Object();
var prevbool = true;
ObjTable.colCount=0;                    //列数
ObjTable.rowCount=0;                    //行数
ObjTable.Map = null;                    //Table Map
ObjTable.prevRow = -1;                  //前一单元格的行号，默认为-1
ObjTable.prevCol = -1;                  //前一单元格的列号，默认为-1
ObjTable.curRow=-1;                     //当前行号，默认为-1
ObjTable.curCol=-1;                     //当前列号，默认为-1
ObjTable.Element="TD";                  //待操作元素的 Tag Name
```

当窗体载入时，将表格的相关信息保存在 ObjTable 对象中，关键代码如下：

```
ObjTable.Load = function(a_sID,a_sTagName){ //载入 table
 var iRowCount=0, iColCount=0, i, j, m, n, iIndex=0, iCount;
 var sTable = document.getElementById(a_sID);
 if (sTable!=null) {                          //设置 table 属性
     var tableMap = [];
     iColCount=sTable.rows[0].cells.length;
     iRowCount = sTable.rows.length;          //获取总行数
     //监听 onClick 事件
     sTable.getElementsByTagName('tbody')[0].onclick=ObjTable_setFocus;
     //监听 onKeyDown 事件
     document.getElementsByTagName('body')[0].onkeydown=ObjTable_moveFocus;
     ObjTable.colCount = iColCount;           //列数
     ObjTable.rowCount = iRowCount;           //行数
     var aCols=null, iCell;
     for (i=0; i<iRowCount; ++i) {
         aCols = new Array(iColCount);
         tableMap.push(aCols);
     }
     for (i=0; i<iRowCount; ++i) {
         iIndex=0;
         for (j=0; j<iColCount; j+=iCell.colSpan) {
             if (tableMap[i][j]==null) {
                 iCell = sTable.rows[i].cells[iIndex++];
                 for(m=i; m<i+iCell.rowSpan; ++m) {
                     for(n=j; n<j+iCell.colSpan; ++n) {tableMap[m][n] = i+','+j;}
                 }
                 tableMap[i][j] = iCell;
             }
         }
     }
     ObjTable.Map = tableMap;                 //table map 结束
   }
};
```

（3）自定义鼠标单击单元格函数 ObjTable_setFocus()，当鼠标单击单元格时，设置单元格的前景色和背景色，关键代码如下：

```
//鼠标单击设置焦点
var iCurRow,iCurCol;
function ObjTable_setFocus(event){
 var e = event || window.event;
 var obj = e.target || e.srcElement, oParent = obj.parentNode;
 var iCurRow = ObjTable.curRow, iCurCol = ObjTable.curCol;
 if (prevbool==true){
     var iCurRow = ObjTable.curRow, iCurCol = ObjTable.curCol;
 }
 else{
     ObjTable.prevRow = iCurRow;ObjTable.prevCol = iCurCol;
 }
 RecallFocus(ObjTable);
```

```
var oPrevNode = ObjTable.getNode(ObjTable.prevRow,ObjTable.prevCol);
var iColCount = ObjTable.colCount;
var aMaps = ObjTable.Map;
if (oParent.tagName.toUpperCase() != "TR") {
        return;
}
iCurRow = oParent.rowIndex;
for (var i=0; i<iColCount; i++){
        if (aMaps[iCurRow][i]==obj) {
                iCurCol = i;
                break;
        }
}
```

（4）自定义函数 ObjTable_moveFocus()，当用键盘的方向键移动单元格焦点时取消上一次的单元格焦点颜色，使当前单元格获得焦点，关键代码如下：

```
//移动焦点
function ObjTable_moveFocus(event) {
//获取当前节点
var e = event || window.event;
var oNode = e.target || e.srcElement;
var oNext = null;
switch (e.keyCode) {
        case 37://左键 2
                oNext = CTable_getNextNode(2);
                break;
        case 38://上键 0
                oNext = CTable_getNextNode(0);
                break;
        case 13://Enter 键
        case 39://右键 3
                oNext = CTable_getNextNode(3);
                break;
        case 40://下键 1
                oNext = CTable_getNextNode(1);
                break;}
//改变背景色
if (oNext) {
                oNext.focus();
                oNext.bgColor="#336699";
                oNext.style.color="#FFFFFF";}
```

（5）自定义函数 CTable_getNextNode()，用于获取下一个单元格节点位置，关键代码如下：

```
//获取下一个节点
function CTable_getNextNode(a_key) {
var oNode = null;
var iCol = ObjTable.curCol ,iRow=ObjTable.curRow;
var iRowCount = ObjTable.rowCount, iColCount=ObjTable.colCount;
var aMaps = ObjTable.Map, aTemps=null;
var iCurCol=iCol, iCurRow = iRow, iTemp;
var sTemp="", sStr="";
oNode = aMaps[iRow][iCol];
if (typeof(oNode)=="string") {
        aTemps = oNode.split(",");
        iCurRow = aTemps[0];
        iCurCol = aTemps[1];
}while (true) {
        switch(a_key) {
                case 0:
                        iRow--;
                        break;
                        case 1:
                        iRow++;
                        break;
                        case 2:
                                iCol--;
                                if (iCol<0) {
                                        iCol = iColCount-1;
                                        iRow--;
```

```
                    }
                break;
                case 3:
                    iCol++;
                    if (iCol>=iColCount) {
                        iRow++;
                        iCol=0;
                    }
                break;
}
if (iRow<0) {
    iRow=iRowCount-1;
}
if (iRow>=iRowCount) {
    iRow=0;
}
if ((iCurRow == iRow) && (iCurCol == iCol)) {
    continue;
}
oNext = aMaps[iRow][iCol];
sTemp=typeof(oNext);
if (sTemp.toLowerCase()=="string") {
    aTemps=oNext.split(",");
    if ((iCurRow!=aTemps[0]) || (iCurCol!=aTemps[1])){break;}
        continue;
    }
    break;
};switch(a_key) {
    case 0://上
    case 1://下
        if (oNext.colSpan>1) {
            ObjTable.prevCol = CTable.curCol;
            ObjTable.prevRow = ObjTable.curRow;
            ObjTable.curRow = iRow;
        }else {
            ObjTable.prevCol = ObjTable.curCol;
            ObjTable.prevRow = ObjTable.curRow;
            ObjTable.curCol = iCol;
            ObjTable.curRow = iRow;}
        break;
    case 2://左
    case 3://右
        if (oNext.rowSpan>1) {
            ObjTable.prevCol = ObjTable.curCol;
            ObjTable.curCol = iCol;
            iTemp = ObjTable.curRow;
            ObjTable.curRow = iRow;
            ObjTable.prevRow = iTemp;
        }else {
            ObjTable.prevCol = ObjTable.curCol;
            ObjTable.prevRow = ObjTable.curRow;
            ObjTable.curCol = iCol;
            ObjTable.curRow = iRow;}
        break;}
RecallFocus(ObjTable);
prevbool=false;
iCurRow=ObjTable.curCol;
iCurCol=ObjTable.curRow;
return oNext;
```

自定义函数 RecallFocus()，在单元格失去焦点时使用单元格的前景色和背景色恢复正常，并当窗体载入时将表格的相关信息保存到自定义对象 ObjTable 中，关键代码如下：

```
function RecallFocus(c){
var oPrev = ObjTable.getNode(c.prevRow,c.prevCol);
if (oPrev) {
    oPrev.bgColor="#FFFFFF";
    oPrev.style.color="#000000";
}
}
```

```
//窗体载入函数
window.onload=function() {ObjTable.Load("table1","TD");};
</script>
```

秘笈心法

通过控制键盘来随意移动单元格焦点时，用到了一个重要的属性 keycode 键盘对应值。本实例在 JavaScript 脚本中定义了 ObjTable_moveFocus()函数，通过 switch()条件语句来设置键盘方向键的上、下、左、右以及 Enter 键的动作。

实例 508　隐藏及显示单元格　初级

光盘位置：光盘\MR\19\508　实用指数：★★★

实例说明

本实例主要通过对指定单元格的隐藏，可以对表格中的说明性文字实现一种动态的显示效果。当鼠标移动到蓝色区域时，光标会变成手状，单击此处左边的注册帮助将进行隐藏和显示的切换效果，程序的运行结果如图 19.18 和图 19.19 所示。



图 19.18　显示单元格

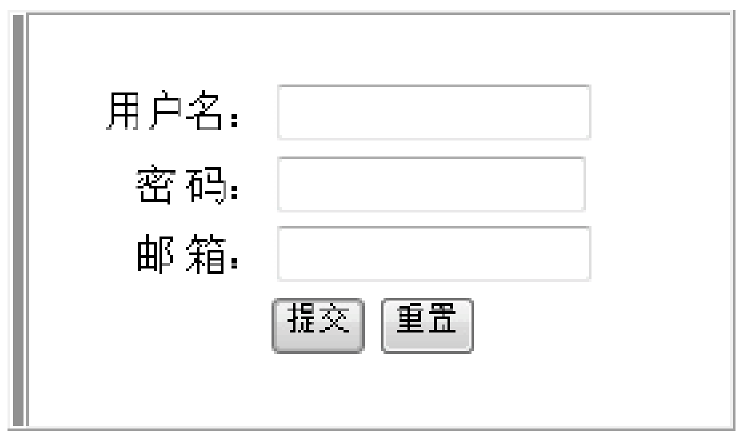


图 19.19　隐藏单元格

关键技术

本实例主要是通过单元格 style 样式中的 display 属性值 none 和 block 来实现当前单元格的隐藏和显示。

当鼠标移入和移出页面的“蓝色区域”时，将使用 document 对象的 body 的 style 的 cursor 属性值来改变光标的形状。cursor 属性值如表 19.1 所示。

表 19.1　Style 的 cursor 属性值

属　性　值	说　　明	属　性　值	说　　明
auto	标准光标	Text	I 形光标
default	标准箭头	Vertical-text	水平 I 形光标
hand	手形光标	No-drop	不可拖动的光标
help	帮助光标	All-scroll	三角方向光标
move	移动光标	Crosshair	十字光标
wait	等待光标	Not-allowed	无效光标

设计过程

（1）创建 index.jsp 页面文件，在页面中添加一个表格及相关属性，关键代码如下：

```
<table id="table1" width="408" height="167" border="1" cellpadding="0" cellspacing="0">
  <tr>
    <td width="119" valign="top" id="tdhide"><p>注册——帮助</p>
    <p>注意在这里填写的一定要是真实资料，以免在丢失后可以方便找回，如果有与色情、法轮功等有违国家相关规定的行为后果自负。
</p></td>
```

```
<td width="1" bordercolor="#FFFFFF" bgcolor="#6495ed" id="tdsize" onClick="hidecell()"
            onMouseMove="move()" onMouseOut="out()"> </td>
    <td width="274" id="tdcen" align="center">
      <table width="257" height="108" border="0" bordercolor="#000000" align="center">
            <Tr>
                  <Td   align="right">用户名：</Td>
                  <td align="left"><input type="text" name="tname" size="15"/></td>
            </Tr>
```

（2）使用 JavaScript 脚本自定义函数 move()，当鼠标移动到蓝色区域时，使光标变成手形状，关键代码如下：

```
function move(){
 document.body.style.cursor="hand";
}
```

自定义函数 out()，当鼠标移出此蓝色区域时，使光标恢复标准形状，关键代码如下：

```
function out(){
 document.body.style.cursor="auto";
}
```

自定义函数 hidecell()，用于控制单元格的隐藏及显示，同时改变表格的大小，关键代码如下：

```
var bool=true;
var td1=0;
function hidecell(){
 if (bool==true){
        tdhide.style.display="none";
        td1=parseInt(tdhide.width);
        table1.width=table1.width-tdhide.width;
        bool=false;
 }
 else{
        tdhide.style.display="block";
        table1.width=parseInt(table1.width)+td1;
        bool=true;
 }
}
</script>
```

秘笈心法

在控制单元格显示与隐藏时，首先定义了一个布尔类型变量 bool，此值为 true 时则显示单元格，反之则隐藏。使单元格的 name 的 style.display 值设置为 none 显示状态，再用 parseInt()方法强制转换成整数，以便求出整个表格的新宽度。

实例 509　**编辑单元格中的文本信息**　初级

光盘位置：光盘\MR\19\509　实用指数：★★★

实例说明

本实例是在表格中选择指定的单元格，单击“修改”按钮，将在按钮的下面显示一个文本框，当对文本框中的信息进行修改时，所选单元格中的文本信息也随之改变，运行结果如图 19.20 所示。

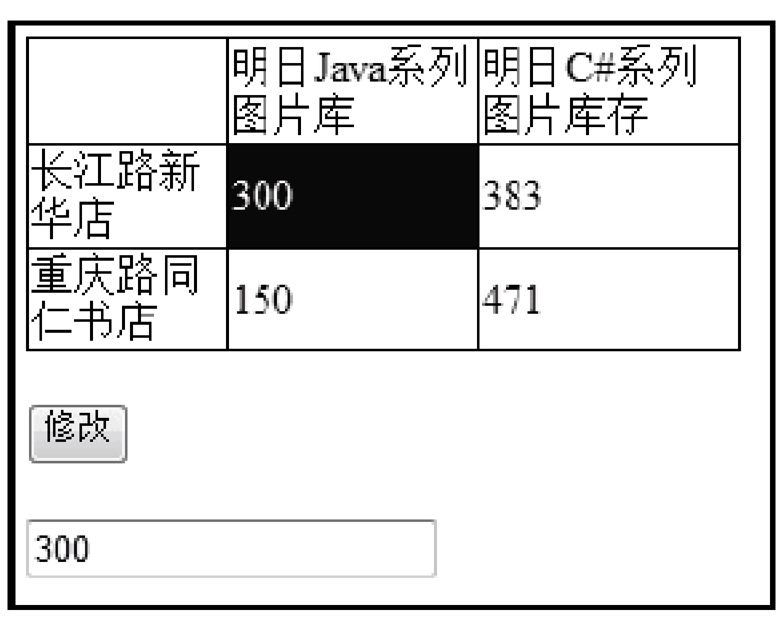


图 19.20　修改后的单元格的效果

关键技术

本实例主要应用 JavaScript 脚本，并使用 setExpression()方法将文本框中的内容动态传递给已选中的单元格，也就是修改表格中的 innerHTML 属性，并在修改其他单元格之前用 removeExpression()方法动态删除 innerHTML 属性。

setExpression()方法的语法格式如下：

```
setExpression(attribute,value,language)
```

参数说明

❶ attribute：动态属性。

❷ value：属性值。

❸ language：可选参数，语言，如 JavaScript。

功能：用于设置动态及相关属性。

removeExpression()方法的语法格式如下：

```
removeExpression(attribute)
```

参数说明

attribute：动态属性。

功能：删除动态属性。

设计过程

（1）创建 index.jsp 页面，在页面中添加一个表格及文本框和按钮，并设置表格的属性名称，关键代码如下：

```
<table width="270" cellspacing="1" bordercolor="#000000" bgcolor="#000000" id="table1">
  <tr bgcolor="#FFFFFF">
    <td id="cell_1"></td>
    <td id="cell_2">明日 Java 系列图片库</td>
    <td id="cell_3">明日 C#系列图片库存</td>
  </tr>
  <tr bgcolor="#FFFFFF">
    <td id="cell_4">长江路新华店</td>
    <td id="cell_5">300</td>
    <td id="cell_6">383</td>
  </tr>
  <tr bgcolor="#FFFFFF">
    <td id="cell_7">重庆路同仁书店</td>
    <td id="cell_8">150</td>
    <td id="cell_9">471</td>
  </tr>
</table>
<script language="javascript">
```

（2）使用 JavaScript 脚本实现修改单元格内容，自定义函数 select()，用来保存发生事件的文档元素信息，关键代码如下：

```
var lastSelection = null;
//获取选择行或单元格的参数值
function select(element) {
 var e, r, c;
 if (element == null){e = window.event.srcElement;}
 else {e = element;}
 if (e.tagName == "TD"){
        c = findCell(e);
        if (c != null){
                if (lastSelection != null) {deselectRowOrCell(lastSelection);}
                selectRowOrCell(c);
                lastSelection = c;
        }
 }
 window.event.cancelBubble = true;
}
```

（3）自定义可修改单元格内容的 editContents()函数，关键代码如下：

```
//编辑单元格中的文本信息
function editContents() {
        var c, p, nr;
        if (lastSelection == null)
                return false;
        c = lastSelection;
        if (c.tagName != "TD") {
                return null;
        }
```

```
        document.form1.celltext.style.display = "";
        document.form1.celltext.value = c.innerHTML;
        c.setExpression("innerHTML", "document.form1.celltext.value");
        document.form1.celltext.focus();
        document.form1.celltext.onblur = unhookContentsExpression;
}
```

（4）编写自定义函数 clearContentsExpression()，用于清空文本框中的内容，关键代码如下：

```
//对文本框进行清空
function clearContentsExpression(){
        lastSelection.removeExpression("innerHTML");
        document.form1.celltext.value = '';
        document.form1.celltext.style.display = "none";
}
</script>
```

秘笈心法

本实例中，在编辑单元格中的文本信息时，关键之处在于获取单元格信息。本实例是在 JavaScript 脚本中应用了 focus()函数获取焦点，如一个表单，当用鼠标单击或双击表单中的文本框时，它可以触发一个事件，以此来实现编辑单元格的文本信息。

实例 510　单元格外边框加粗

光盘位置：光盘\MR\19\510

初级　实用指数：★★★

实例说明

本实例是对单元格外边框的粗细进行设置，在实际开发中可以起到美观和突出显示相应表格的作用，程序的运行结果如图 19.21 所示。



图 19.21　单元格外边框加粗效果

关键技术

本实例主要是通过设置<table>表格的 border 属性来改变表格的外边框粗细的效果。修改表格外边框宽度，首先是在 table 表格中添加 border 属性然后设置其值为大于 1，数值越大外边框就越粗，相反则越细。

设计过程

创建 index.jsp 页面，在页面中添加一个表格，并设置相应的 id 名称，关键代码如下：

```
<table   align="center" border="5" bordercolor="#6495ed">
   <tr>
      <td   id="Td1"> </td>
      <td   id="Td2">body 子对象</td>
      <td   id="Td3">属性</td>
   </tr>
   <tr>
      <td id="Td3">名称 1</td>
      <td id="Td5">tr</td>
      <td id="Td6">表格行</td>
   </tr>
   <tr>
      <td id="Td7">名称 2</td>
      <td id="Td8">td</td>
      <td id="Td9">表格列</td>
   </tr>
</table>
```

秘笈心法

对于外边框的操作在实际开发中多数用于突出显示某个重要单元格信息。例如，在用户注册页面时可以对验证码的外边框进行加粗并给予颜色变色。

19.3　表格的特殊效果

在制作网页时，最常用到的就是表格属性，可以使用表格来设计页面的布局，也可以将表格设计成不同的风格使其页面更加美观。

实例 511	闪烁的表格边框 光盘位置：光盘\MR\19\511	初级 实用指数：★★★

实例说明

本实例是将表格的边框设置为指定的宽度，然后在页面载入时按一定的时间间隔改变表格的外边框颜色，运行结果如图 19.22 所示。



图 19.22　表格边框闪烁的效果

关键技术

本实例主要在 JavaScript 脚本中设置一个颜色的 array()集合，然后使用 for()语句循环这个集合，并设置表格边框颜色等于这个集合的顺序排列的值，最后应用 setTimeout()方法预设一段时间修改表格的外边框颜色。修改表格的外边框颜色是用表格的 style 对象的 bordercolor 属性来实现的。

设计过程

（1）创建 index.jsp 页面，并在页面中添加一个表格，设置其外边框的宽度为 5，这样可以更直观地看到表格变色的效果，关键代码如下：

```
<body>
<table id="table1" border=5>
<tr><td align=center><strong>心锐志远，明日词典</strong></td>
</tr>
<tr><td align=center>
<pre></pre>
</table>
```

（2）利用 JavaScript 脚本编写实现表格边框变色的关键代码如下：

```
<script language="javascript">
var i=0;
var Color= new Array("#1e90ff","#66cdaa","#d2691e","#c71585","#483d8b");
function tt(){
        if (i>Color.length-1){
                i=0
        }
        table1.style.borderColor=Color[i];
        i=i+1;
        setTimeout("tt()",100);
}
tt();
</script>
```

秘笈心法

在 HTML 语言中一些对象属性中间是带“-”字符的，如 border-color，要想在脚本中进行应用，应将“-”符号删除，并将“-”字符后的首字母大写（borderColor）。

实例 512　选中行的变色

光盘位置：光盘\MR\19\512

初级　实用指数：★★★

实例说明

本实例是实现当鼠标指向表格中的任意一个单元格时，将该单元格所在行的背景颜色及字体颜色进行改变，运行结果如图 19.23 所示。

	文科平均成绩	理科平均成绩
一班	90	88
二班	92	87.5

图 19.23　选中行的变色效果

关键技术

实现本实例，主要是当鼠标指向表中的单元格时，将通过 onMouseOver 事件调用自定义函数 over()来改变单元格所在行的前景色和背景色，当鼠标纵向离开单元格时将通过 onMouseOut 事件将所选行的前景色和背景色改变为初始状态。

设计过程

（1）创建 index.jsp 页面，在页面中添加一个表格并设置相应的 id 名称，关键代码如下：

```
<table align="center"    border="1">
  <tr id="tr1" onMouseOver="over(this.id)" onMouseOut="out(this.id)">
    <td> </td>
    <td>文科平均成绩</td>
    <td>理科平均成绩</td>
  </tr>
  <tr id="tr2" onMouseOver="over(this.id)" onMouseOut="out(this.id)">
    <td>一班</td>
    <td>90</td>
    <td>88</td>
  </tr>
  <tr id="tr3" onMouseOver="over(this.id)" onMouseOut="out(this.id)">
    <td>二班</td>
    <td>92</td>
    <td>87.5</td>
  </tr>
</table>
```

（2）使用 JavaScript 脚本实现选中行变色的效果，关键代码如下：

```
<script language="javascript">
function over(trname){
        eval(trname).style.backgroundColor="5f9ea0";
        eval(trname).style.color="FFFFFF";
}
function out(trname){
        eval(trname).style.backgroundColor="FFFFFF";
        eval(trname).style.color="000000";
}
</script>
```

秘笈心法

本实例在 JavaScript 脚本中应用了 eval()函数，当鼠标单击单元格时变换一个单元格行的背景颜色，而当焦点离开时再使用 eval()函数把单元格行的颜色恢复为初始状态。eval()函数用于把一个字符串当作一个 JavaScript 表达式执行。

实例 513　表格中表元内部空白

光盘位置：光盘\MR\19\513

初级　实用指数：★★★

实例说明

在网站的显示数据页面中，往往因为数据过多而在显示时过于密集，导致查询者眼花缭乱。本实例介绍如

何改变表格中表元内部的空白，使表格数据看起来更加清晰，运行结果如图 19.24 和图 19.25 所示。

Food	Drink	Sweet
A	B	C

图 19.24　未设置表元内部空白的效果

Food	Drink	Sweet
A	B	C

图 19.25　设置表元内部空白的效果

关键技术

本实例主要是设置了表格的 cellpadding 属性来实现表元间距，cellpadding 属性的值越大则表元的间隙越大，相反则越小，默认为 0。cellpadding 属性只能在<table>标记中添加，不可以在其下属性中添加，如<tr/>和<td/>。

设计过程

创建 index.jsp 页面，在该页设置表格的相关元素，关键代码如下：

```
<table border="1"  cellpadding="5">
        <Tr>
                <td>用户名:</td>
                <td><input type="text" name="uname" size="20"/></td>
        </Tr>
        <Tr>
                <td>密 码:</td>
                <td><input type="password" name="pwd" size="21" /></td>
        </Tr>
        <Tr>
                <td><input type="submit" name="Submit" value="登录" /></td>
        </Tr>
</table>
```

秘笈心法

cellpadding 属性在实际开发中多数设置为 2 或者默认，对于 cellpadding 属性也可以在 CSS 样式表中设置，然后在页面中调用即可。

实例 514　表格中表元间隙

光盘位置：光盘\MR\19\514

初级

实用指数：★★★

实例说明

对于表格与表格之间的设置实例 513 只是其中的一种，还有一种是对于表格中显示大量数据的间隙设置。本实例实现对表格中表元间隙的设置，运行结果如图 19.26 和图 19.27 所示。

表元间隙设置：

Food	Drink	Sweet
A	B	C

图 19.26　未设置表元间隙效果

表元间隙设置：

Food	Drink	Sweet
A	B	C

图 19.27　设置表元间隙效果

关键技术

本实例通过设置<table>表格的 cellspacing 属性来改变表元内间隙的效果，同样 cellspacing 属性的值越大，其表元间隙间距越大，相反其值越小间隙越小，默认为 1。

设计过程

创建 index.jsp 页面，在该页面中添加表格的相关元素及属性，关键代码如下：

```
<table border cellspacing=#>表元间隙设置:
  <table border cellspacing="12">
  <tr><th>Food</th><th>Drink</th><th>Sweet</th>
  <tr><td>A</td><td>B</td><td>C</td>
  </table>
```

秘笈心法

本实例中用到了 table 的 cellspacing 属性，其实它就是在原有小格的基础上加上 10 个单位宽度，也就是把原来的小格变大，但原来写的内容占的大小范围不变。

实例 515　对表格内文字进行对齐

光盘位置：光盘\MR\19\515

初级　实用指数：★★★

实例说明

在网站中显示数据时，为了整体效果的统一，开发者会对其设置居中、左对齐或右对齐。本实例实现的就是把表格的文字分别设置为左对齐、居中、右对齐的效果，运行结果如图 19.28 所示。



图 19.28　对表格内文字设置不同的对齐效果

关键技术

本实例主要应用表格的 align 属性的 3 个值，分别是 left（左对齐）、center（居中）和 right（右对齐）。这 3 个值可以任意应用在<table>中的某一位置，可以添加在<tr>或<td>中。通常情况下会设置为居中效果使整体效果更加醒目。

设计过程

创建 index.jsp 页面，在该页面中添加表格的相关元素及属性，关键代码如下：

```
<h3>表格内文字的对齐</h3>
 <table border width=160>
  <tr>
 <th>apple</th><th>banana</th><th>orange</th>
 </tr>
 <Tr>
    <td align=left>A</td>
    <td align=center>B</td>
    <td align=right>C</td>
    </Tr>
 </table>
```

秘笈心法

在使用<table>的 align 属性时要小心，不是所有的浏览器都能识别它。如果要让表格显示在屏幕中央，使用<center>标识符来包含表格会更安全些。

实例 516　对表格内信息进行布局

光盘位置：光盘\MR\19\516

初级　实用指数：★★★

实例说明

在网站显示信息时，也有相应的格式布局，在实际开发中，客户可能会要求将表格内的文字统一向上或居中对齐。本实例是把 4 个表格的内容分别设置成顶端位置、居中位置、底端位置和基线位置，运行结果如图 19.29 所示。

One	Two	Three	Four
A	B	C	D

图 19.29　表格内布局效果

关键技术

本实例在实现表格中的信息布局效果时，用到了 valign 属性的 4 个值，语法格式如下：

```
tabledataObject.vAlign=top|middle|bottom|baseline
```

功能：设置或返回单元格内数据的垂直排列方式。

其中，valign 的值 top、middle、bottom、baseline，分别表示顶端对齐、居中对齐、底端对齐、基线对齐。

设计过程

创建 index.jsp 页面，在该页面添加表格的相关元素及属性，关键代码如下：

```
<table border height=100>
    <tr>
        <th>One</th><th>Two</th>
        <th>Three</th><th>Four</th>
    </tr>
    <tr>
          <td valign=top>A</td>
          <td valign=middle>B</td>
          <td valign=bottom>C</td>
          <td valign=baseline>D</td>
      </tr>
</table>
```

秘笈心法

在本实例中，设置表格内布局用到了 valign 属性，它与 align 看起来十分相似，用法也一样，它们的区别在于 valign 是垂直位置的操控，align 是水平位置的设置。

实例 517　对表格的大小进行设置

光盘位置：光盘\MR\19\517

初级

实用指数：★★★

实例说明

在网站开发中，经常会遇到填写数据时超出表格的预设大小，这时就需要开发者对可输入表格的大小进行设置，预设置表格的最大宽度和最大高度。本实例就是设置两个预设大小表格的高度和宽度，即 150×150 和 80×80 两个表格，运行结果如图 19.30 所示。



图 19.30　预设表格的高度和宽度

关键技术

本实例应用了<table>表格的高度（height）和宽度（width）来预设表格的大小。设置时可以只设高度或宽度，如果设置高度为 150，那么当表格中的内容达到最大时将自动增加表格宽度；相反如果设置宽度为 150，那么当表格内的信息达到最大时将自动增加高度。

设计过程

创建 index.jsp 页面，并设置两个表格大小分别为 150×150 和 80×80，关键代码如下：

```
  <table><tr><td>
    <table border="1" bgcolor="#6495ed" width="150" height="150">
      <tr>
```

```
            <Td>这个是 150*150 的表格</Td>
        </tr>
    </table>
    <table border="1" bgcolor="lightblue" width="80" height="80" >
        <tr>
            <td>这个是 80*80 的表格</td>
        </tr>
    </table>
    </td></tr>
    </table>
```

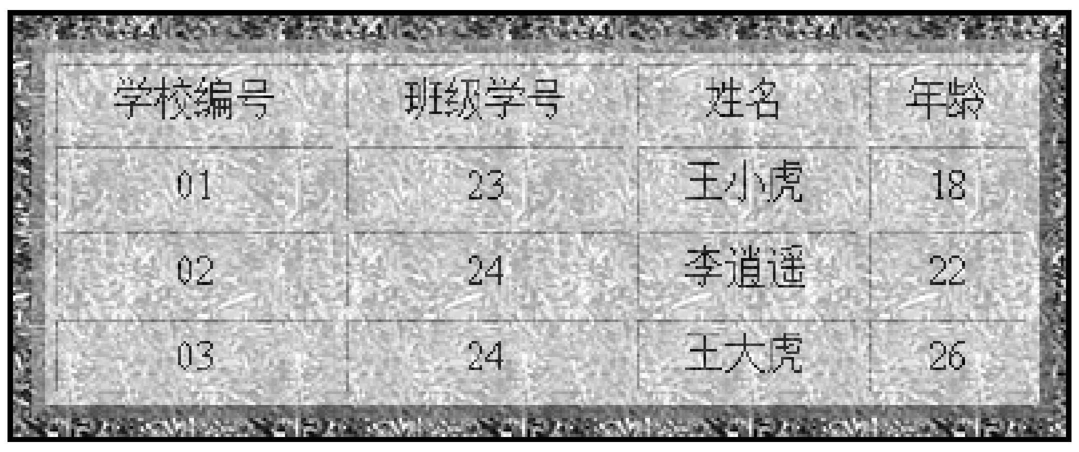
秘笈心法

设置<table>表格的 width 和 height 大小时，除了可输入具体的数字以外，还可以以百分比的形式设置，不过这种情况下一般是在多个表格中应用，当包含在里面的表格各宽度设置为 50%时，是按被包含表格的百分比设置的。

实例 518	透明表格 光盘位置：光盘\MR\19\518	初级 实用指数：★★★

实例说明

在制作网页时，有时需要将表格进行透明处理，以更加突出要显示的部分。本实例是通过设置表格滤镜中的透明度来实现该效果的，运行结果如图 19.31 所示。



学校编号	班级学号	姓名	年龄
01	23	王小虎	18
02	24	李逍遥	22
03	24	王大虎	26

图 19.31　透明效果的图片

关键技术

本实例主要是通过设置表格的 style 的 filter 的 alpha 滤镜的 opacity 值（opacity 的取值范围为 0～100），来实现表格的透明效果的。

设计过程

（1）创建 index.jsp 页面文件，并设置表格及相应的 id 名称，关键代码如下：

```
<table width="400" border=5 align=center cellpadding="4" cellspacing="4" bgcolor="#FFFFFF" id="table1" style="border-color:#FFCCFF;">
  <tr>
    <td id="btb" align=center>学校编号</td>
    <td id="btb" align=center>班级学号</td>
    <td id="btb" align=center>姓名</td>
    <td id="btb" align=center>年龄</td>
  </tr>
  <tr>
    <td id="btb" align=center>01</td>
    <td id="btb" align=center>23</td>
    <td id="btb" align=center>王小虎</td>
    <td id="btb" align=center>18</td>
  </tr>
  <tr>
    <td id="btb" align=center>02</td>
    <td id="btb" align=center>24</td>
    <td id="btb" align=center>李逍遥</td>
    <td id="btb" align=center>22</td>
  </tr>
  <tr>
    <td id="btb" align=center>03</td>
    <td id="btb" align=center>24</td>
    <td id="btb" align=center>王大虎</td>
    <td id="btb" align=center>26</td>
  </tr>
</table>
```

（2）使用 JavaScript 脚本实现表格透明效果，关键代码如下：

```
<script language="javascript">
table1.style.filter="alpha(opacity=40)";
</script>
```

秘笈心法

本例中用到的 alpha 滤镜的常用属性如下。

- enabled：可选参数，布尔类型，设置或检索滤镜是否激活。
- style：整数值，设置或检索透明渐变的样式，分别为 0、1、2、3。0：默认值，整体透明度将 Opacity 值均匀作用于对象。1：线性渐变透明度，从由 startX 和 startY 点决定的透明度开始，到由 finishX 和 finishY 决定的点，由 FinishOpacity 点结束。2：圆形放射渐变透明度。圆形放射区域的圆心为对象的中心，圆周到与对象的边相切为止。透明渐变由圆心开始，到圆周结束。开始透明度由 Opacity 决定，结束透明度由 FinishOpacity 决定。3：矩形放射渐变透明度。由对象的边开始，到对象的中心结束。开始透明度由 Opacity 决定，结束透明度由 FinishOpacity 决定。

实例 519 限制表格的宽度

光盘位置：光盘\MR\19\519

初级

实用指数：★★★

实例说明

在向表格中输入信息时，当信息大于单元格设置的宽度时，单元格的宽度将自动向右扩展，这样使表格看上去极不美观。本实例将限制单元格的宽度，当信息超出单元格的宽度时，单元格的高度将增大，运行结果如图 19.32 所示。

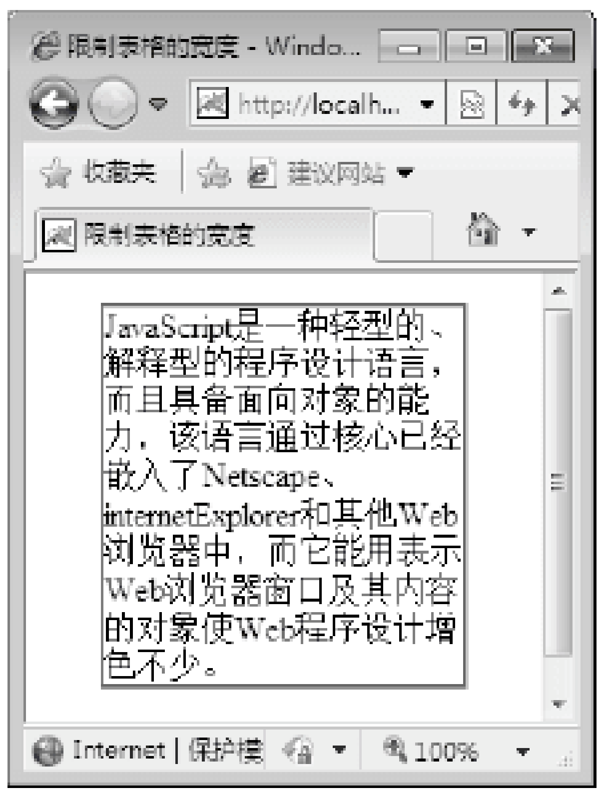


图 19.32　限制表格的宽度

关键技术

本实例主要应用表格的 style 样式的 tableLayout 属性的 fixed 值来控制表格的大小，在用 tableLayout 属性之前，要用表格的 width 和 height 属性来设置表格的宽度和高度。下面对 style 样式的 tableLayout 属性进行详细说明。

tableLayout 属性的语法格式如下：

```
tableLayout:auto|fixed
```

参数说明

❶ auto：默认的自动算法（计算速度慢）布局将基于各单元格的内容，表格的每一个单元格读取计算之后才会显示出来。

❷ fixed：固定布局的算法。在这种算法中，水平布局是仅基于表格的宽度、表格边框的宽度、单元格间距、列的宽度，和表格内容无关。

设计过程

（1）创建 index.jsp 页面，在页面中添加一个表格并设置表格的 id 名称，关键代码如下：

```
<table id="table1" border="1px" cellpadding="0px" cellspacing="0px" bordercolor="#000000"
 bordercolordark="#6495ed" bordercolorlight="#666666"><tr>
      <td>JavaScript 是一种轻型的、解释型的程序设计语言，而且具备面向对象的能力，该语言通过核心已经嵌入到了 Netscape、
      internetExplorer 和其他 Web 浏览器中，而它能用表示 Web 浏览器窗口及其内容的对象使 Web 程序设计增色不少。</td>
   </tr>
</table>
```

（2）利用 JavaScript 编写用于固定单元格宽度的关键代码如下：

```
<script language="javascript">
         var w=180;
         var h=6;
         table1.align="Center";
         table1.style.width=w;
         table1.style.height=h;
```

```
	table1.style.tableLayout="fixed";
</script>
```

秘笈心法

在本实例中，在 JavaScript 中用表格的 style 样式来获取其宽度和高度，并把自定义变量 w 和 h 分别赋值给宽度和高度，再使用 tableLayout 的 fixed 属性来固定表格。

实例 520	表格的标题 光盘位置：光盘\MR\19\520	初级 实用指数：★★★

实例说明

在某些网站中，当对某些图片或数据进行操作处理时，由于页面过于复杂，很难一下子找到要操作的对象，这时就需要使用一个标题将其括起来，并进行文字性说明提示，本实例将用表格来实现这一效果，运行结果如图 19.33 所示。



图 19.33　表格的标题

关键技术

本实例主要应用<fieldset>和<legend>标记将表单元素组合在一起，形成一个表格标题，并用层的 innerHTML 属性将设置好的表的标题以动态形式添加到层中。下面对<fieldset>和<legend>标记进行说明。

- <fieldset>标记：对表单中的元素进行分组。
- <legend>标记：用来标记子表单。

设计过程

（1）创建 index.jsp 页面，在页面中添加一个层，并设置相关的属性，关键代码如下：

```
<div id="Tdiv" align="center" width="200px"></div>
```

（2）使用 JavaScript 编写用于显示表格标题的关键代码如下：

```
<script language="javascript">
function caption(name){
	var c="<table width='260px' height='180' align='center'>";
	c=c+"<tr><td>";
	c=c+"<fieldset><legend><font>"+name+"</font></legend>";
	c=c+"<img src='photo.jpg' width='200' height='140'></fieldset>"
	c=c+"</td></tr></table>";
	Tdiv.innerHTML=c;
}
</script>
```

秘笈心法

本实例在 JavaScript 脚本中定义了一个变量 c，然后使用此变量输出一个<table>表格及其相关的表单元素内容，并设置表单的高为 180px、宽为 260px，并且使用 align 属性设置居中效果。

实例 521	表格的外阴影 光盘位置：光盘\MR\19\521	初级 实用指数：★★★

实例说明

本实例为了能够让表格呈现立体效果，在表格的外边框、单元格及文本的后面都加了阴影效果，实例运行

结果如图 19.34 所示。



图 19.34　表格的外阴影

关键技术

本实例主要应用表格的 style 样式的 filter 和 shadow 属性来对表格进行阴影效果设置。下面对 shadow 属性进行详细说明。

shadow 属性的语法格式如下：

```
shadow(color,direction,strength)
```

参数说明

❶ color：阴影的颜色设置。

❷ direction：设置阴影的显示角度。

❸ strength：设置阴影的长度。

功能：对当前组件进行阴影设置。

设计过程

（1）创建 index.jsp 页面文件，在页面中添加一个表格及相关的属性，关键代码如下：

```
<table id="table1" style="border-color:#0000FF" border=5>
<tr><td align=center><strong>放飞梦想，与翼翱翔</strong></td>
</tr>
<tr><td align=center style="font-size: x-large">
<pre>明日科技</pre>
</table>
```

（2）使用 JavaScript 脚本用于实现表格阴影效果，关键代码如下：

```
<script langrige="javascript">
table1.style.filter="progid:DXImageTransform.Microsoft.Shadow(Color=#6954ed,Direction=220,strength=10)";
</script>
```

秘笈心法

本实例在 JavaScript 脚本中应用了 table 表格的 style 样式和 filter 过滤样式，在 JavaScript 中主要是通过对表格的外边框进行设置来改变显示效果，并且对所有的表格生效，这与 CSS 样式的设置表格效果有些类似。

实例 522　立体表格

光盘位置：光盘\MR\19\522

初级　实用指数：★★★

实例说明

在浏览网页时，有些网页中的表格行标记都具有立体效果，本实例将实现这一效果，实例运行结果如图 19.35 所示。

	JavaWeb	C#	VB
2010年	5000本	4500本	3700本
2009年	5800本	5500本	4000本
2008年	6800本	6500本	5300本

图 19.35　立体效果表格

关键技术

本实例主要是对表格的 borderColorLight（边框的灯光颜色）、borderColorDark（边缘模糊颜色）和 bgColor 属性进行颜色设置，并主要设置每行的单元格的 bgColor 属性，然后通过 for()语句来循环设置表头和表列颜色变换的效果。

设计过程

（1）创建 index.jsp 页面，在页面中添加一个表格并设置相应的标记和属性，关键代码如下：

```
<table id="table1" >
  <tr id="b_tr" >
    <td align=center > </td>
    <td align=center nowrap >JavaWeb</td>
```

```
        <td align=center nowrap >C#</td>
        <td align=center nowrap >VB</td>
    </tr>
    <tr>
        <td align=center nowrap id="tb1">2010 年</td>
        <td align=center nowrap >5000 本</td>
        <td align=center nowrap >4500 本</td>
        <td align=center nowrap >3700 本</td>
    </tr>
    <tr>
        <td align=center nowrap id="tb2">2009 年</td>
        <td align=center nowrap >5800 本</td>
        <td align=center nowrap >5500 本</td>
        <td align=center nowrap >4000 本</td>
    </tr>
    <tr>
        <td align=center nowrap id="tb3">2008 年</tds>
        <td align=center nowrap >6800 本</td>
        <td align=center nowrap >6500 本</td>
        <td align=center nowrap >5300 本</td>
    </tr>
</table>
```

（2）使用 JavaScript 脚本实现立体表格效果，关键代码如下：

```
<script language="javascript">
        table1.border=1;
        table1.cellSpacing=0;
        table1.borderColorLight="#333333";
        table1.borderColorDark="#efefef";
        b_tr.bgColor="#cccccc";
        var s="";
        for (var i=1;i<4;i++){
                s=eval("tb"+i).id;
                eval(s).bgColor="#cccccc";
}
</script>
```

秘笈心法

在本实例中，设置 borderColorLight 和 borderColorDark 属性在窗体要呈现立体感时，需要两个边框为亮色，剩下两个边框为暗色。此例中还用到 eval 函数，它可以把一个字符串当作一个 JavaScript 表达式一样去执行，简单地说，eval 函数接收一个参数 S，如果 S 不是字符串，则直接返回 S 语句，如果 S 语句执行的结果是一个值，则返回此值，否则返回 underfined。

实例 523	虚线边框表格 光盘位置：光盘\MR\19\523	初级 实用指数：★★★

实例说明

在用表格输出数据时，可以对表格的边框及单元格边框的样式进行设置，这样可以避免表格的单调性，浏览起来不会乏味。本实例将表格的边框设置为虚线，运行结果如图 19.36 所示。

	名字	专业	工作年龄
01	王小虎	信息科学自动化	4年
02	李小虎	电汽与机修	6年
03	钱小晓	金融管理	3年

图 19.36　虚线边框表格

关键技术

本实例主要应用单元格的 style 样式的 borderTop、borderBottom、borderLeft 和 borderRight 属性来设置单元格的上、下、左、右 4 条边框的颜色、粗细及样式。首先，把表格的外边框及单元格边框设置为 0，然后再使用单元格的 style 样式中的 borderTop 和 borderLeft 属性，将所有单元格的上边和左边以虚线显示，接下来将右边最后一列的单元格用 style 样式中的 borderRight 属性将右边以虚线显

示，最后把最后一行的单元格用 style 样式中的 borderBottom 属性将下边以虚线显示。

设计过程

（1）创建 index.jsp 页面，在页面中添加一个表格并设置表格相应的属性，然后将表格的外边框宽度设为 0，关键代码如下：

```
<table id="table1" align=center width="300" border="0" cellspacing="0">
  <tr>
    <td id="bt" align=center> </td>
    <td id="bt" align=center>名字</td>
    <td id="bt" align=center>专业</td>
    <td id="bt" align=center>工作年龄</td>
  </tr>
  <tr>
    <td id="bt" align=center>01</td>
    <td id="bt" align=center>王小虎</td>
    <td id="bt" align=center>信息科学自动化</td>
    <td id="bt" align=center>4 年</td>
  </tr>
  <tr>
    <td id="bt" align=center>02</td>
    <td id="bt" align=center>李小虎</td>
    <td id="bt" align=center>电汽与机修</td>
    <td id="bt" align=center>6 年</td>
  </tr>
  <tr>
    <td id="bt" align=center>03</td>
    <td id="bt" align=center>钱小晓</td>
    <td id="bt" align=center>金融管理</td>
    <td id="bt" align=center>3 年</td>
  </tr>
</table>
```

（2）使用 JavaScript 编写实现虚线边框表格的关键代码如下：

```
<script language="javascript">
var i=0;
var row=table1.rows.length;
var column=table1.rows(1).cells.length;
for (i=0;i<row*column;i++){
 bt[i].style.borderTop="#000000   1px   dotted";
 bt[i].style.borderLeft="#000000   1px   dotted";
 for (var j=1;j<=column;j++)
        bt[j*row-1].style.borderRight="#000000   1px   dotted";
 if (i>=(row-1)*column)
        bt[i].style.borderBottom="#000000   1px   dotted";
}
</script>
```

秘笈心法

本实例中用到的 style 样式的 borderTop 为设置对象上边框的样式。例如，设置 border-top:thin 等于设置 border-top:thin none，而 border-top-color 的默认值将采用文本颜色。因此此前的任何 border-top-color 和 border-top-width 设置都会被清除。

实例 524	表格作为分割线 光盘位置：光盘\MR\19\524	初级 实用指数：★★★

实例说明

在浏览网页时，经常会看到网页上用一些分割线将网页分为几个部分，使用户浏览页面时更容易找到相应的信息。本实例将用表格来作为页面的分割线，并且在表格中以渐变颜色进行动态显示，运行结果如图 19.37 所示。

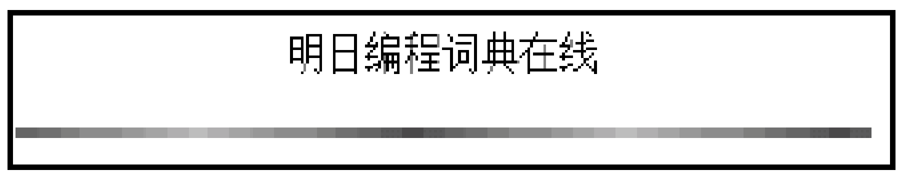


图 19.37　表格作为分割线

关键技术

本实例要使用 document 对象的 write()方法在页面中动态添加一个 1 行 40 列的表格，并将表格外边框及单元格的边框设置为 0，最后使用 setTimeout()方法以指定时间动态修改每个单元格的背景颜色来实现表格的动态渐变效果。

设计过程

创建 index.jsp 页面，在页面中填写“明日编程词典在线”，然后在其下面使用 JavaScript 脚本实现表格作为分割线的效果，关键代码如下：

```
<p align="center">明日编程词典在线</p>
<script language="javascript">
        changcol=Array(6,7,8,9,'a','b','b','c','d','e','f')
        t="<table height=4 width=100% cellspacing=0 cellpadding=0><tr>"
        for(x=0;x<40;x++){
                t+="<td id=td"+x+"></td>";
        }
        document.write(t+"</tr></table>");
        function line(y){
                for(i=0;i<40;i++){
                        c=(i+y)%20;
                        if(c>10){
                                c=20-c;
                        }
                        document.all["td"+(i)].bgColor="#00"+changcol[c]+"00"+changcol[c]+"";
                }
                y++;
                setTimeout('line('+y+')',1);
        }
        line(1);
</script>
```

秘笈心法

本实例中使用 document 对象的 write()方法向页面输出一个表格，其中的 cellspacing 属性是表元间隙设置；cellpadding 是表元内部空白设置。再创建一个实例属性 document.all[]，用于表格行中的颜色变换。

实例 525	表格向下展开 光盘位置：光盘\MR\19\525	初级 实用指数：★★★

实例说明

本实例将给表格赋予一种动态的展开效果，当鼠标移动到表格中时，表格将自动向下展开，当鼠标移出表格时，表格将恢复到原来的样子，运行结果如图 19.38 和图 19.39 所示。

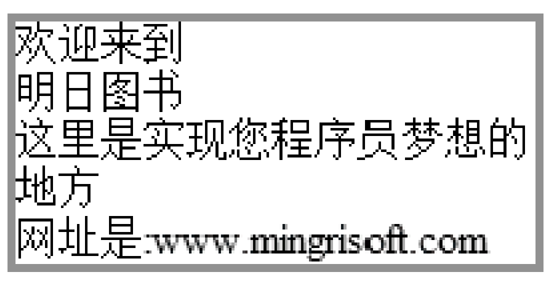


图 19.38　表格展开之前的效果



图 19.39　表格展开之后的效果

关键技术

本实例主要是用 setTimeout()方法在表格的 onMouseOver（鼠标移入）和 onMouseOut（鼠标移出）事件中

以指定的时间动态增加或减小表格的高度（height），当高度增加或减少到一定程度时，用 clearTimeout()方法将 setTimeout()的值进行清除。

设计过程

（1）创建 index.jsp 页面，在页面中添加一个表格，并设置表格的 id 名称及相关属性，关键代码如下：

```
<table width="200" height="30" border="2" cellpadding="0" cellspacing="0" id="mytd"
onMouseOver="over()" onMouseOut="out()" bordercolor="#6495ed">
 <tr>
        <td>欢迎来到<br>明日图书<br>这里是实现您程序员梦想的地方
        <br>网址是:www.mingrisoft.com</td>
 </tr>
</table>
```

（2）使用 JavaScript 编写用于实现表格的向下展开效果，自定义函数 over()，表示将表格的高度自动增加，关键代码如下：

```
<script language="JavaScript">
var act;
function over(){
 var h = parseInt(mytd.height);
 if (h < 120){
        mytd.height = h + 2;
        clearTimeout(act);
        act = setTimeout('over()', 10);
 }
}
```

（3）自定义函数 out()，表示将增加高度后的表格恢复到原来的高度，关键代码如下：

```
function out(){
 var h = parseInt(mytd.height);
 if (h > 30){
        mytd.height = h - 2;
        clearTimeout(act);
        act = setTimeout('out()', 10);
 }
}
</script>
```

秘笈心法

本实例使用了 parseInt()函数把表格的高度转换成整数，如果高度小于 120，表格的高度就增加 2。parseInt()函数用于返回字符串转换得到的整数，语法格式如下：

```
parseInt(numString,[radix])
```

其中，numString 为要转换的字符串，radix 为可选参数，在 2～36 之间的表示 numString 所保存数字进制的值。

实例 526　表格向右拉伸

光盘位置：光盘\MR\19\526

初级

实用指数：★★★

实例说明

本实例是实现表格一种动态的拉伸效果，在鼠标光标移动到表格内时，表格将自动向右拉伸，而当光标移出表格时，表格又恢复原来的样子，运行结果如图 19.40 和图 19.41 所示。



图 19.40　鼠标未移入表格内时



图 19.41　鼠标移入到表格内时

关键技术

本实例主要应用 setTimeout()方法在 table 表格的属性中添加鼠标移入事件 onMouseOver 和鼠标移出事件 onMouseOut，并以指定时间动态地增减表格的宽度（width），当宽度增加到一定程度时，则使用 clearTimeout()方法将 setTimeout()方法的值清除。

设计过程

（1）创建 index.jsp 页面文件，在页面中添加一个表格并设置相应的 id 名称及其属性，关键代码如下：

```
<table width="150" height="30" border="2" cellpadding="0" cellspacing="0" id="mytd"
onMouseOver="over()" onMouseOut="out()" bordercolor="#dcdcdc">
<tr>
        <td>这是一个鼠标<br>移入到表格内<br>便拉长表格宽度的效果
        <br>详细请登录:www.mingrisoft.com</td>
</tr>
</table>
```

（2）使用 JavaScript 脚本实现表格向右拉伸效果，自定义函数 over()和 out()，用于实现将表格宽度自动增加和把增加宽度后的表格恢复到原来的宽度，关键代码如下：

```
<script language="JavaScript">
var act;
function over(){
 var w = parseInt(mytd.width);
 if (w< 180){
        mytd.width = w + 50;
        clearTimeout(act);
        act = setTimeout('over()', 10);
 }
}
function out(){
 var w = parseInt(mytd.width);
 if (w > 150){
        mytd.width = w - 50;
        clearTimeout(act);
        act = setTimeout('out()', 10);
 }
}
</script>
```

秘笈心法

本实例中用到了 clearTimeout()方法，用于终止 setTimeout()方法的执行，语法格式如下：

```
clearTimeout(ID_of_settimeout)
```

参数说明

ID_of_settimeout：为由 setTimeout()返回的 ID 值，该值标识要取消的延迟执行代码块。

第 5 篇

操作 Word、Excel、报表与打印篇

⏭ 第 20 章　JSP 操作 Word

⏭ 第 21 章　JSP 操作 Excel

⏭ 第 22 章　报表与打印

第20章

JSP 操作 Word

- 应用 JavaScript 导出到 Word
- 应用响应流导出到 Word
- 应用 POI 组件导出到 Word

20.1　应用 JavaScript 导出到 Word

实例 527　将 JSP 页面的信息在 Word 中打开

光盘位置：光盘\MR\20\527

高级

实用指数：★★★★

实例说明

在开发动态网站时，经常会遇到打印页面中指定表格的情况，这时可以将要打印的表格导出到 Word 中，然后再打印。本实例将介绍如何将页面中的订单列表导出到 Word 中并打印。运行本实例，在页面中将显示订单信息列表，单击“打印”超链接后，将把 Web 页中的数据导出到 Word 的新建文档中，并保存在 Word 的默认文档保存路径中，最后调用打印机打印该文档。实例运行效果如图 20.1 和图 20.2 所示。



图 20.1　刚加载页面之后

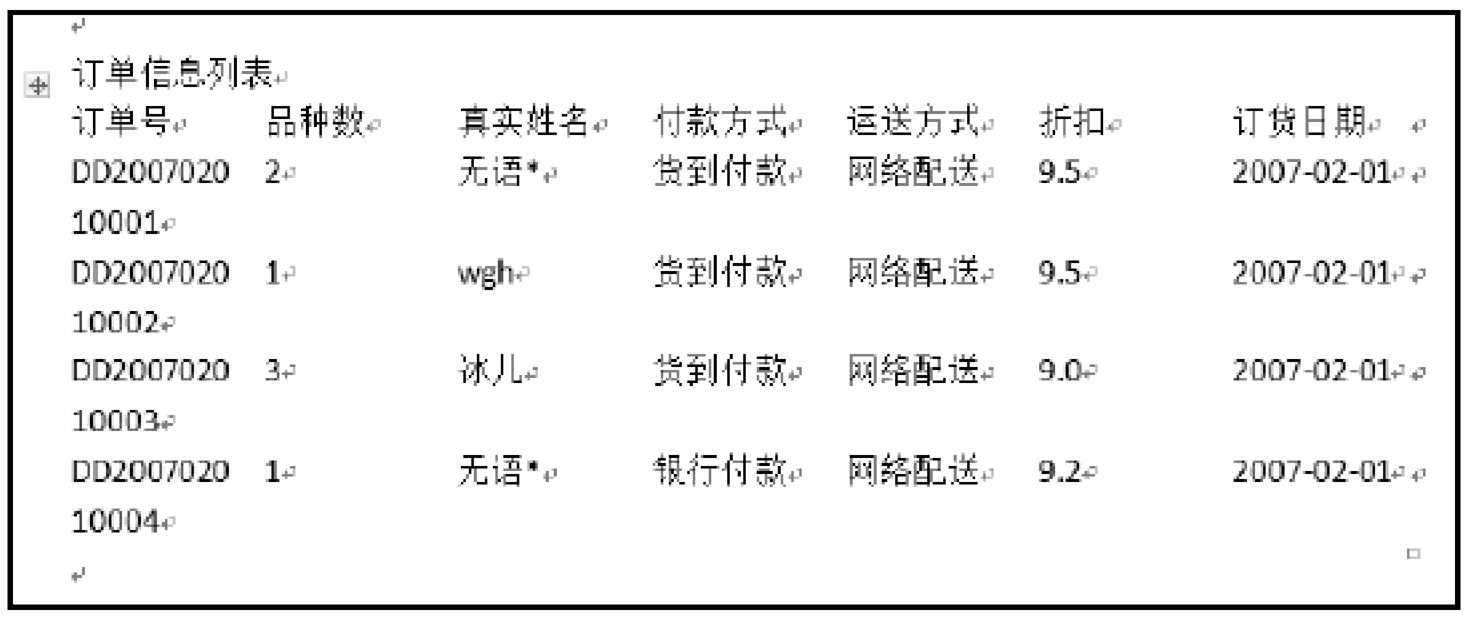


图 20.2　保存到 Word 中

关键技术

本实例主要应用 JavaScript 的 ActiveXObject()构造函数创建一个 OLE Automation(ActiveX)对象的实例，并应用该实例的相关方法实现。

ActiveXObject()构造函数的一般语法格式如下：

```
var objectVar = new ActiveXObject(class[, servername]);
```

参数说明

❶ objectVar：用于指定引用对象的变量。

❷ class：用于指定应用程序的名字或包含对象的库，并且指定要创建的对象的类型。采用 library.object 的语法格式，如 Word.Application 表明要创建的是 Word 对象。

❸ servername：可选参数，用于指定包含对象的网络服务器的名字。

设计过程

（1）将显示订单信息的表格的 id 设置为 order，因为要打印该表格中的数据。关键代码如下：

```
<table  id="order" width="100%" height="48"  border="1" cellpadding="0" cellspacing="0" bordercolor="#FFFFFF" bordercolordark="#CCCCCC"
bordercolorlight="#FFFFFF">
```

（2）编写自定义 JavaScript 函数 outDoc()，用于将 Web 页面中的订单信息导出到 Word 中，并进行自动打印，代码如下：

```
<script language="javascript">
function outDoc(){
  var table=document.all.order;
  row=table.rows.length;
  column=table.rows(1).cells.length;
```

```
    var wdapp=new ActiveXObject("Word.Application");
    wdapp.visible=true;
    wddoc=wdapp.Documents.Add();   //添加新的文档
    thearray=new Array();
//将页面中表格的内容存放在数组中
 for(i=0;i<row;i++){
 thearray[i]=new Array();
 for(j=0;j<column;j++){
              thearray[i][j]=table.rows(i).cells(j).innerHTML;
 }
 }
var range = wddoc.Range(0,0);
range.Text="订单信息列表"+"\n";
wdapp.Application.Activedocument.Paragraphs.Add(range);
wdapp.Application.Activedocument.Paragraphs.Add();
rngcurrent=wdapp.Application.Activedocument.Paragraphs(3).Range;

var objTable=wddoc.Tables.Add(rngcurrent,row,column)        //插入表格
for(i=0;i<row;i++){
 for(j=0;j<column;j++){
 objTable.Cell(i+1,j+1).Range.Text = thearray[i][j].replace(" ","");
 }
}
wdapp.Application.ActiveDocument.SaveAs("orderInfo.doc",0,false,"",true,"",false,false,false,false,false);
wdapp.Application.Printout();
wdapp=null;
}
</script>
```

（3）通过单击“打印”超链接调用自定义的 JavaScript 函数 outDoc()，关键代码如下：

```
<a href="#" onClick="outDoc();">打印 </a>
```

秘笈心法

在 Word 中查看并修改默认文档保存路径的方法如下：选择“工具”→“选项”命令，在弹出的对话框中选择“文件位置”选项卡，然后选中“文档”列表项，单击“更改”按钮，在弹出的对话框中选择默认文档保存路径，连续单击“确定”按钮即可。

20.2 应用响应流导出到 Word

实例 528	将表单数据输出到 Word 中 光盘位置：光盘\MR\20\528	高级 实用指数：★★★★

实例说明

在网页中寻找需要的资料或信息时，有时想把页面中的内容复制下来，然后再存放到文档中进行修改和编辑，无形之中就多了一个程序复制过程，本实例将网页中表单元素提交的数据输出到 Word 文档中，单击“提交”按钮，即可将表单中的数据输出到 Word 中，运行结果如图 20.3 和图 20.4 所示。



图 20.3　在页面中输入的信息



图 20.4　将网站的表单元素输入到 Word 中

关键技术

本实例把页面中表单 method 属性设置为 post 提交方式，而在相应的 ReferServletWord.java 中将 Post()方法提交的数据转换到 get()方法中执行。ReferServetWord 类继承了 HttpServlet 类，使用 setContentType()方法来设置以 Word 文档为表现形式，然后声明一个打印文本输出流 out，使用 out.println()方法和 request.getParameter()方法来获取表单中相关的元素信息并打印在文档中。

设计过程

（1）创建添加用户信息的 index.jsp 页面，将表单中的数据提交到 Servlet 类的 ReferServletWord 中，关键代码如下：

```
<form name="form" method="post" action="ReferServletWord" onSubmit="return checkEmpty(form)">
  <table width="276" border="1" cellpadding="0" cellspacing="0">
    <tr align="center">
      <td width="67" height="30">姓名</td>
      <td width="203"><input type="text" name="name"></td>
    </tr>
    <tr align="center">
      <td height="30">性别</td>
      <td><input type="text" name="sex"></td>
    </tr>
    <tr align="center">
      <td height="30">年龄</td>
      <td><input type="text" name="age"></td>
    </tr>
    <tr align="center">
      <td height="30">出生</td>
      <td><input type="text" name="born"></td>
    </tr>
    <tr align="center">
      <td height="30">职业</td>
      <td><input type="text" name="profession"></td>
    </tr>
  </table>
<br>
    <inpCut type="submit" name="Submit" value="提交">
  </form>
```

（2）创建 ReferServletWord.java 类，该类继承 HttpServlet 类，在 Form 表单中主要触发 ReferServletWord 类，自动执行 doPost()和 doGet()方法，关键代码如下：

```
public void doGet(HttpServletRequest request, HttpServletResponse response) throws
            ServletException, IOException {
        response.setContentType("application/msword");    //设置为 Word 格式
        PrintWriter out = response.getWriter();             //声明 out 打印实例
        out.println(request.getParameter("name"));          //获取表单中的 name
        out.println(request.getParameter("sex"));
        out.println(request.getParameter("age"));
        out.println(request.getParameter("born"));
        out.println(request.getParameter("profession"));
    }
    public void doPost(HttpServletRequest request, HttpServletResponse response) throws
            ServletException, IOException {
        doGet(request, response);
    }
}
```

（3）在 web.xml 文件中配置 ReferServletWord 类的代码如下：

```
<servlet>
    <servlet-name>ReferServletWord</servlet-name>
    <servlet-class>com.pkh.servlet.ReferServletWord</servlet-class>
</servlet>
<servlet-mapping>
     <servlet-name>ReferServletWord</servlet-name>
     <url-pattern>/ReferServletWord</url-pattern>
</servlet-mapping>
```

秘笈心法

本实例主要应用 HttpServletResponse 对象的 setContentType()方法设置文件的表现形式，其关键代码如下：

```
response.setContentType("application/msword");
```

SetContentType 头提供了响应文档的 MIME（Multipupose Internet Mail Extension，多用途 Internet 邮件扩展）类型。通过 Response.setContentType()语句来设置 HTTP 的响应头，返回固定格式的页面。

实例 529　将查询结果输出到 Word 中

光盘位置：光盘\MR\20\529

高级

实用指数：★★★★

实例说明

网站上提供的信息很全面，如果能将需要的内容直接保存到 Word 中，将会为浏览者提供极大的方便。本实例便实现了这样的功能，运行程序，如图 20.5 所示，选择要查询的专业名称，单击“查询”按钮，查询结果将自动以 Word 文档显示到 JSP 页面上，如图 20.6 所示。



图 20.5　查询页面信息

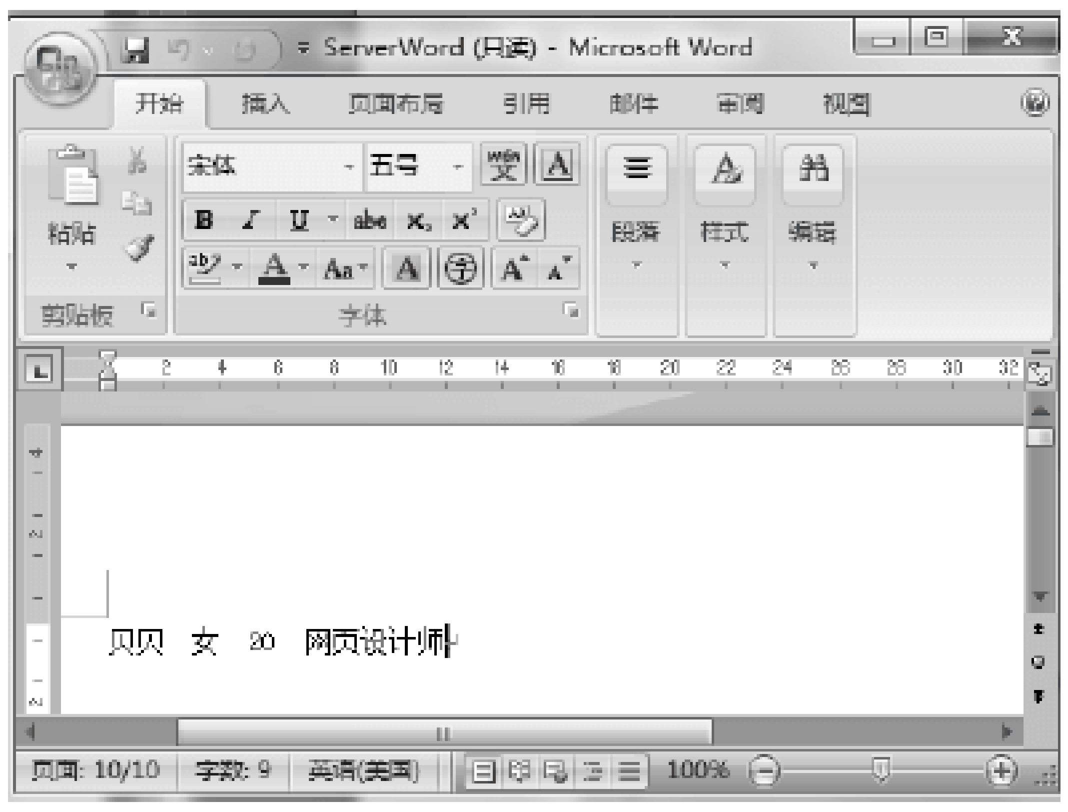


图 20.6　将查询结果保存到 Word 中

关键技术

本实例主要应用 HttpServletResponse 类的 setContentType()方法设置文件的表现形式，因本例是从数据库中查询出信息并将查询结果显示到 Word 中，所以将 index.jsp 页面中的下拉列表框的 name 属性设置为 profession，在 ServerWord 类中获取到查询条件，代码如下：

```
String profession = request.getParameter("profession");
```

然后实例化数据库类的操作并执行 executeQuery(sql)方法，最后使用 while()语句循环输出结果集。

设计过程

（1）创建 index.jsp 页面，在页面中添加 Form 表单，在下拉列表框中输出查询条件，关键代码如下：

```
<%
  UserDao connection = new UserDao();
  ResultSet rs = connection.executeQuery("select distinct profession   from tb_word");
%>
  <body>
  <div align="center">
      <br>
      <table width="222" height="225" border="0" cellpadding="0" cellspacing="0">
        <tr>
          <td background="leftsearch.gif"><div align="center">按专业查询
        </div>
            <form name="form" method="post" action="ServerWord">
        <table width="222" height="34" border="0" cellpadding="0" cellspacing="0">
            <tr align="center">
              <td width="174" height="32">
```

```
                <select name="profession">
                  <%
                try {
                  while (rs.next()) {
            %>
                  <option value="<%=rs.getString("profession")%>"><%=rs.getString("profession")%> </option>
                  <%}} catch (SQLException e) {}
           connection.closeConnection();
          %>
                </select>
            </td>
            <td width="148">
                <input type="submit" name="Submit2" value="查询">
            </td>
```

（2）创建 UserDao 类，定义数据库的连接查询和关闭的方法，关键代码如下：

```
public class UserDao {
      private final String url = "jdbc:mysql://localhost:3306/db_database16";
      private final String userName = "root";
      private final String password = "111";
      private Connection con = null;
      private Statement stmt = null;
      private ResultSet rs = null;
      public UserDao() {   //通过构造方法加载数据库驱动
        try {
              Class.forName("com.mysql.jdbc.Driver").newInstance();
        } catch (Exception ex) {
              System.out.println("数据库加载失败");
        }
      }
      //创建数据库连接
      public boolean creatConnection() {
            try {
                  con = DriverManager.getConnection(url, userName, password);
                  con.setAutoCommit(true);
            } catch (SQLException e) {
                  System.out.println(e.getMessage());
                  System.out.println("creatConnectionError!");
            }
            return true;
      }
      //对数据库的查询操作
        public ResultSet executeQuery(String sql) {
            ResultSet rs;
            try {
                  if (con == null) {
                        creatConnection();
                  }
                  Statement stmt = con.createStatement();
                  try {
                        rs = stmt.executeQuery(sql);
                  } catch (SQLException e) {
                        System.out.println(e.getMessage());
                        return null;
                  }
            } catch (SQLException e) {
                  System.out.println(e.getMessage());
                  System.out.println("executeQueryError!");
                  return null;
            }
            return rs;
      }
    }
```

（3）创建 ServerWord.java 类，该类继承 HttpServlet 类，主要执行 doPost()和 doGet()方法，在 Form 表单中触发 ServerWord 类中的方法，关键代码如下：

```
public void doGet(HttpServletRequest request, HttpServletResponse response) throws
            ServletException, IOException {
      response.setContentType("application/msword");
```

```
        String profession = request.getParameter("profession");
        PrintWriter out = response.getWriter();
        String sql = "select * from tb_word where profession='" + profession +"'";
        UserDao connection = new UserDao();
        ResultSet rs = connection.executeQuery(sql);
        try {
            while (rs.next()) {
                out.print(rs.getString("name"));
                out.print("    ");
                out.print(rs.getString("sex"));
                out.print("    ");
                out.print(rs.getString("age"));
                out.print("    ");
                out.print(rs.getString("profession"));
                out.println();
            }
        } catch (SQLException ex) {
        }
        connection.closeConnection();
}
```

秘笈心法

本实例比实例 528 复杂的是从数据库中读取信息并显示在页面中，然后再保存到 Word 文档，多了一个查询语句的方法，所以在页面中设置下拉列表框的 name 属性来标记，从而实现从数据库中取读信息，代码如下：

```
<select name="profession">
    <%try {while (rs.next()) {%>
            <option value="<%=rs.getString("profession")%>"><%=rs.getString("profession")%> </option>
        <%}} catch (SQLException e) {}
            connection.closeConnection();
        %>
</select>
```

其中在 while()语句中使用 rs.next 来循环数据库中所有的下拉列表信息，并使用 rs.getString()方法获取。

实例 530　将页面中的学生表以 Word 表格保存

光盘位置：光盘\MR\20\530

高级

实用指数：★★★★

实例说明

在浏览网站时有时想保存其中一个页面，但又不想打开 Word 文档，本实例的操作就是为这个特点编写的，运行本实例。在加载后有一个学生表信息在页面中，单击“导出文档”按钮，弹出相应的保存界面，效果如图 20.7 所示。



图 20.7　以 Word 文档保存的界面

关键技术

本实例在实现以 Word 文档保存时应用了 contentType 属性，设置为 application/msword 即为 Word 文档格式，并用 StringBuffer 语句把要导入的信息保存在内存中，页面的信息头设置代码如下：

```
response.setHeader("Content-Disposition", "attachment; filename=" + recommendedName + "\"");
```

最后应用输出流 ServletOutputStream 输出保存在内存中的信息。

设计过程

（1）创建 index.jsp 页面，在该页面中添加一个学生表的表格以及相关元素，并将 form 表单中的 method 属性使用 post()方法提交，其关键代码如下：

```
<form action="ExportWordServlet" method="post">
<span id="table">
<table bgcolor="#EEECF2" bordercolor="#A3B2CC" border="1" cellspacing="0">
        <tr><th>学号</th><th>姓名</th><th>科目</th><th>分数</th></tr>
        <tr><td>10001</td><td>赵二</td><td>高数</td><td>82</td></tr>
        <tr><td>10002</td><td>张三</td><td>高数</td><td>94</td></tr>
        <tr><td>10003</td><td>李四</td><td>高数</td><td>69</td></tr>
        <tr><td>10001</td><td>赵二</td><td>线数</td><td>77</td></tr>
        <tr><td>10002</td><td>张三</td><td>线数</td><td>61</td></tr>
        <tr><td>10003</td><td>李四</td><td>线数</td><td>87</td></tr>
</table>
</span><br/>
<input type="submit" name="Word" value="导出文档" onclick="test()"/>
     <input type="hidden" id="tableInfo" name="tableInfo" value=""/>
</form>
```

（2）编写 ExportWordServlet 类，在其中的 doPost()方法中实现导出文档的方法，设置 contentType 为 application/msword，即以 Word 文档格式导出，并利用输入/输出流导出文件，其关键代码如下：

```
        String fileName ="导出数据";                             //设置导出的文件名称
        StringBuffer sb=null;
        sb = new StringBuffer(request.getParameter("tableInfo"));       //将表格信息放入内存
        String contentType = "application/mswrod";               //设置导出文件的格式
        String recommendedName = new String(fileName.getBytes(),"utf-8");
        response.setContentType(contentType);
        response.setHeader("Content-Disposition", "attachment; filename=" + recommendedName + "\"");
        response.resetBuffer();
        //利用输入/输出流导出文件
        ServletOutputStream sos = response.getOutputStream();
        sos.write(sb.toString().getBytes());
        sos.flush();
        sos.close();
}
```

（3）在 web.xml 中配置 ExportWordServlet 类，关键代码如下：

```
<servlet>
      <servlet-name>ExportWordServlet</servlet-name>
      <servlet-class>ExportWordServlet</servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>ExportWordServlet</servlet-name>
      <url-pattern>/ExportWordServlet</url-pattern>
   </servlet-mapping>
```

秘笈心法

在 JSP 中应用 Word 操作时，由于微软对 Word 的源代码没有公开，所以在操作 Word 时多数情况会使用 POI 或 JACOB 组件，前者应用 Excel 多一些，相对来说 JACOB 组件应用 Word 更灵活一些，只是它只能运行在 Windows 平台下。

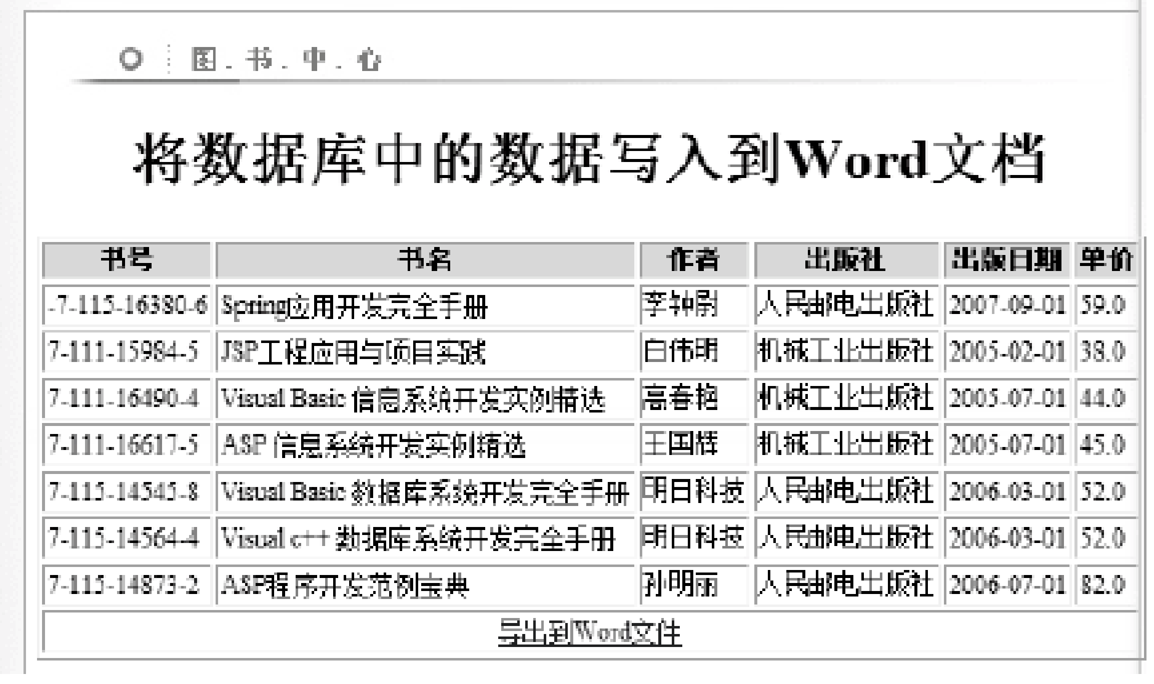
20.3　应用 POI 组件导出到 Word

实例 531　**将数据库中的数据写入到 Word 中**　**高级**

光盘位置：光盘\MR\20\531　　实用指数：★★★★

实例说明

数据库表的生成方式有很多，有使用饼形图、折线图、柱形图等比较直观方式的，也有使用各种表格的，但是考虑报表的修改与其他需要调整的因素，最好为报表实现导出到 Excel 或 Word 文档的功能，这样可以把数据保存成工作文档，更加方便用户的使用。本实例使用 POI 开源组件实现将数据库导出到 Word 文档中，程序运行结果如图 20.8 所示，页面中表格的数据是从数据库读取的图书信息，单击表单下方的“导出到 Word 文件”超链接，后台程序会重新读取数据库中的数据，并生成 Word 文档，如图 20.9 所示。


图.书.中.心

将数据库中的数据写入到Word文档

书号	书名	作者	出版社	出版日期	单价
-7-115-16380-6	Spring应用开发完全手册	李钟尉	人民邮电出版社	2007-09-01	59.0
7-111-15984-5	JSP工程应用与项目实践	白伟明	机械工业出版社	2005-02-01	38.0
7-111-16490-4	Visual Basic 信息系统开发实例精选	高春艳	机械工业出版社	2005-07-01	44.0
7-111-16617-5	ASP 信息系统开发实例精选	王国辉	机械工业出版社	2005-07-01	45.0
7-115-14545-8	Visual Basic 数据库系统开发完全手册	明日科技	人民邮电出版社	2006-03-01	52.0
7-115-14564-4	Visual c++ 数据库系统开发完全手册	明日科技	人民邮电出版社	2006-03-01	52.0
7-115-14873-2	ASP程序开发范例宝典	孙明丽	人民邮电出版社	2006-07-01	82.0

导出到Word文件

图 20.8　程序运行页面前

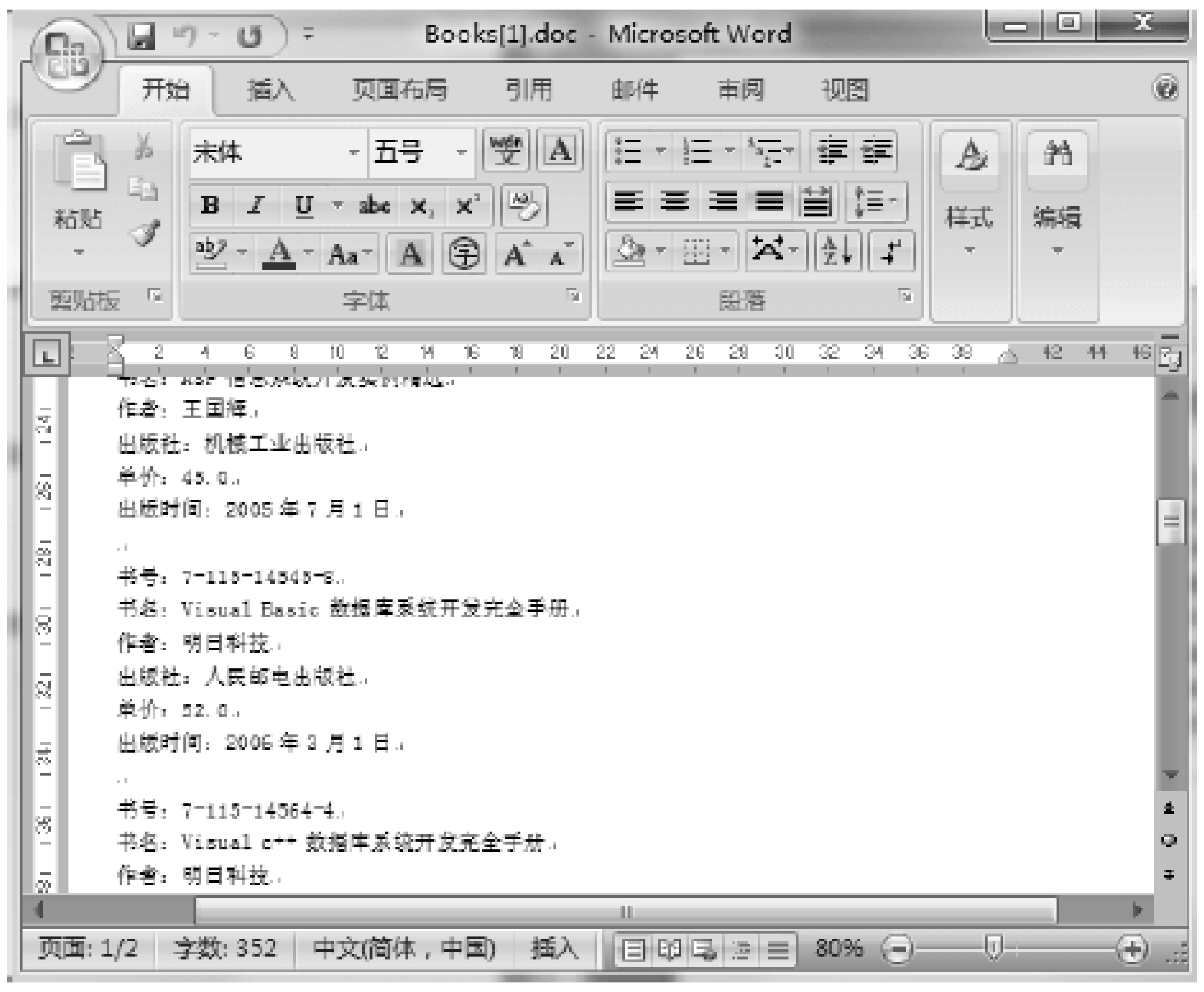


图 20.9　生成的 Word 文档

关键技术

本实例的技术要点都来自于 POI 组件，它是本实例的核心组件，所有与 Word 文档相关的操作都是由 POI 组件完成的，下面介绍实例中使用 POI 组件的关键方法。

（1）获取 DirectoryEntry 目录对象。POIFSFileSystem 是 POI 文件系统，它的实例对象的 getRoot()方法可以获取 DirectoryEntry 目录对象，该对象用于管理 POI 文件系统，其语法格式如下：

```
Public DirectoryNode getRoot()
```

该方法返回值的类型是 DirectoryNode，它是 DirectoryEntry 接口的实现类。

本实例的实例代码首先创建了 POIFSFileSystem 类的实例对象，然后通过该对象获取 DirectoryEntry 对象，关键代码如下：

```
POIFSFileSystem fs = new POIFSFileSystem();          //创建 POI 的文件系统对象
DirectoryEntry directory = fs.getRoot();             //获取目录对象
```

（2）创建文档对象。要读写 Word 文档，就必须创建相应的 DocumentEntry 文档对象，DocumentEntry 只是一个文档接口，DocumentNode 是它的一个实现类，createDocument()方法可以创建该实现类的实例对象，其语法格式如下：

```
Public DocumentEntry createDocument(String name,inputStream stream throws java.io.IOException)
```

参数说明

❶ name：该参数是文档的名称。

❷ stream：该输入流用于创建文档内容。

设计过程

（1）编写 BooksToWordService 类，此类是本实例的业务处理层，它实现数据读取和保存 Word 文档的业务，该类的构造方法用于实例化 BookDao 类的实例对象，并在该类中定义 getBooks()方法，用于获取用户所有图书信息的数据，其关键代码如下：

```
public class BooksToWordService {
    private BookDao dao;
    public BooksToWordService() {
        dao = new BookDao();
    }
    public List getBooks() {
        List books = new ArrayList();
        try {
            books = dao.getBooks();
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        return books;
    }
```

（2）在 BooksToWordService 类中定义 downloadWordDoc()方法，该方法首先获取所有图书信息的数据集合，然后遍历数据集合，将所有图书信息添加到一个 StringBuilder 类的实例对象中，该对象是字符串生成器，最后该字符串生成器中的字符串将传递给 writeWordDoc()方法生成 Word 文档，其关键代码如下：

```
public void downloadWordDoc(OutputStream out) {
    try {
        List books = getBooks();                              //获取所有图书信息
        StringBuilder sb = new StringBuilder();               //创建字符串生成器
        for (Object obj : books) {                            //遍历图书数据集合
            BookBean book = (BookBean) obj;
            sb.append("书号：");                              //添加图书信息到字符串生成器
            sb.append(book.getISBN() + "\n");
            sb.append("书名：");
            sb.append(book.getBookName() + "\n");
            sb.append("作者：");
            sb.append(book.getWriter() + "\n");
            sb.append("出版社：");
            sb.append(book.getPublishing() + "\n");
            sb.append("单价：");
            sb.append(book.getPrice() + "\n");
            sb.append("出版时间：");
            Date date = book.getDate();
            //创建日期格式化对象
            DateFormat df = DateFormat.getDateInstance(DateFormat.LONG);
            sb.append(df.format(date) + "\n");
            sb.append("\n");
        }
        writeWordDoc(sb.toString(),out);                      //将图书信息写入到输出流
    } catch (Exception ex) {
        Logger.getLogger(getClass().getName()).log(Level.SEVERE, null, ex);
    }
}
```

（3）在 BooksToWordService 类中定义 writeWordDoc()方法，它用于创建 Word 文档并将 content 参数中的字符串保存为 Word 文档，然后输出到 ostream 参数指定的输出流中，其关键代码如下：

```
 private void writeWordDoc(String content,OutputStream ostream) {
        ByteArrayInputStream bais = null;                     //创建字节输入流对象
        try {
            byte[] b = content.getBytes();                    //获取字符串的字节数组
            bais = new ByteArrayInputStream(b);               //初始化字节输入流对象
            POIFSFileSystem fs = new POIFSFileSystem();       //创建 POI 的文件系统对象
            DirectoryEntry directory = fs.getRoot();          //获取目录对象
                                                              //创建文档对象
            DocumentEntry de = directory.createDocument("WordDocument", bais);
            fs.writeFilesystem(ostream);                      //把数据写入到输出流
        } catch (IOException e) {
            e.printStackTrace();
```

```
        } finally {                                          //释放资源
            try {
                if (bais != null) {
                    bais.close();
                }
                if (ostream != null) {
                    ostream.close();
                }
            } catch (IOException ex) {
                Logger.getLogger(BooksToWordService.class.getName()).log(Level.SEVERE, null, ex);
            }
```

（4）创建 index.jsp 页面，在该页面中创建 BooksToWordService 业务类的实例对象，调用该对象的 getBooks()方法获取所有图书信息的数据集合，然后遍历该数据集合，在表格中显示所有图书信息，并在表格下方添加“导出到 Word 文件”超链接，其关键代码如下：

```
<table border="1">
    <thead style="background-color:#B5E1E2">
        <tr>
            <th>书号</th>
            <th>书名</th>
            <th>作者</th>
            <th>出版社</th>
            <th>出版日期</th>
            <th>单价</th>
        </tr>
    </thead>
    <tbody>
        <%
for (int i = 0; i < list.size(); i++) {
    BookBean book = (BookBean) list.get(i);
            %>
        <tr>
            <td><%=book.getISBN()%></td>
            <td><%=book.getBookName()%></td>
            <td><%=book.getWriter()%></td>
            <td><%=book.getPublishing()%></td>
            <td><%=book.getDate()%></td>
            <td><%=book.getPrice()%></td>
        </tr>
        <%}%>
        <tr>
            <td colspan="6" align="center">
                <a href="down.jsp">导出到 Word 文件</a>
            </td>
        </tr>
    </tbody>
</table>
```

（5）创建 down.jsp 文件，设置应答对象的内容类型为 application/octet-stream，设置头信息 Content-Disposition 的值为 attachment;filename=download.xls，然后创建 BooksToWordService 的实例对象，调用该对象的 downloadWordDoc()方法，这样就会在客户端出现下载窗口，其关键代码如下：

```
<body>
    <%
    out.clear();
    out = pageContext.pushBody();
    response.setContentType("application/octet-stream");
    response.setHeader("Content-Disposition", "attachment;filename=Books.doc");
    OutputStream outs = response.getOutputStream();
    BooksToWordService service = new BooksToWordService();
    service.downloadWordDoc(outs);
    %>
</body>
```

秘笈心法

本实例首先创建了字节输入流，该输入流包含了所有图书信息，然后依次创建 POIFSFileSystem、DirectoryEntry 和 DocumentEntry 对象。POI 组件是本实例的核心组件，使用 POI 下的 POIFSFileSystem 来创建文件系统对象，并用 DirectoryEntry 来获取目录对象，最后将文档内容写入指定的输出流。

第21章

JSP 操作 Excel

- 应用 JXL 组件操作 Excel
- 应用 POI 组件操作 Excel

21.1　应用 JXL 组件操作 Excel

在软件开发过程中，应用程序对办公软件的操作越来越多，尤其是操作微软的 Excel 电子表格，经常需要将一些数据库数据导出 Excel 文件中，或者将一个 Excel 文件数据读取到数据库中。本节将介绍如何应用开源的 JXL（Java Excel API）组件来操作 Excel，下面通过一些实例介绍 JXL 组件的用法。

实例 532　创建 Excel 工作表

光盘位置：光盘\MR\21\532

初级　实用指数：★★★

实例说明

JXL 是一个韩国程序员开发的操作 Excel 的 Java 类库，应用 JXL 可以方便地操作 Excel 电子表格。JXL 支持 Excel 95 到 Excel 2000 的所有版本，能够生成 Excel 2000 标准格式，支持字体、数字、日期操作，能够修饰单元格属性。本实例首先介绍如何应用 JXL 创建 Excel 工作表，程序运行结果如图 21.1 所示。

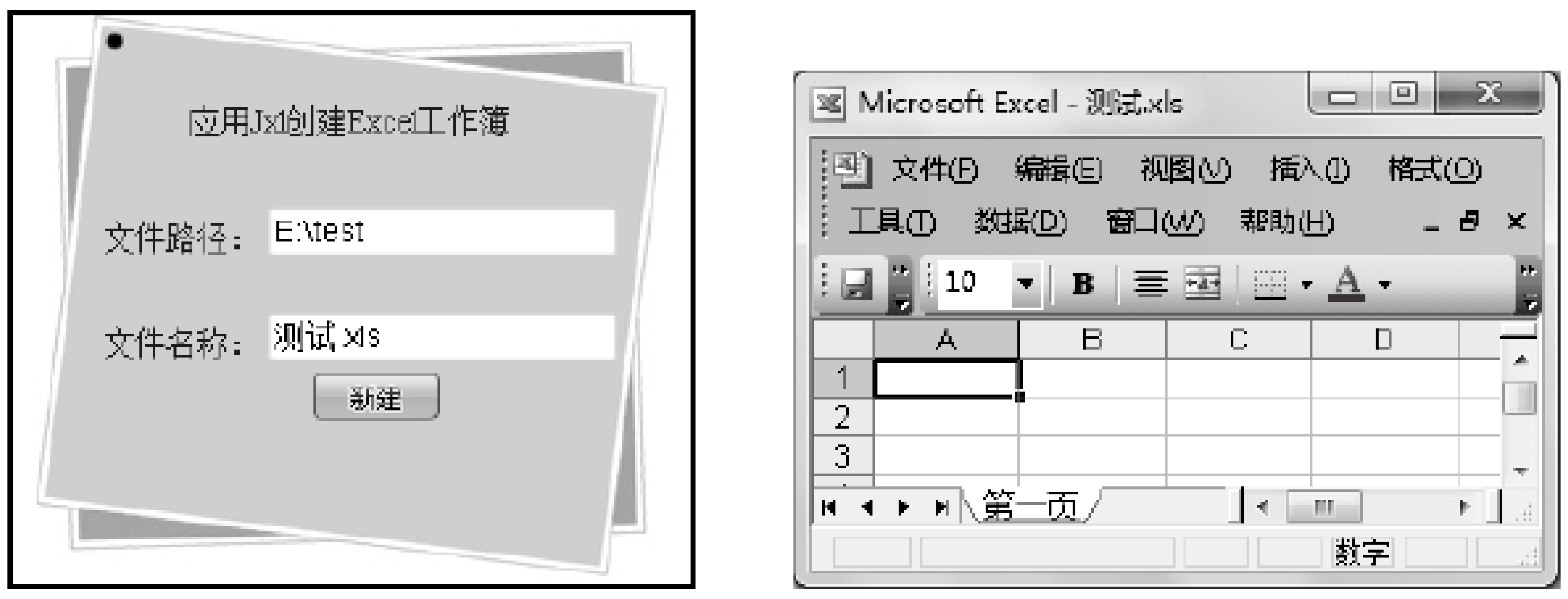


图 21.1　创建 Excel 工作表

关键技术

应用 JXL 组件创建 Excel 工作表之前，需要应用 jxl.Workbook 类的 createWorkbook()方法来创建一个 Excel 工作簿，然后再由工作簿对象的 createSheet()方法创建工作表。下面分别介绍本实例用到的类和方法。

（1）Workbook 类

该类是一个抽象类，创建该类的实例，表示创建一个 Excel 工作簿。该类包含一个用于创建工作簿的静态方法 createWorkbook()，语法结构如下：

```
public static WritableWorkbook createWorkbook(java.io.File file) throws IOException
```

参数说明

file：表示将要创建的 Excel 文件对象。

功能：

创建 Excel 工作簿。

（2）WritableWorkbook 类

该类是一个抽象类，该类的实例表示工作簿对象。在向 Excel 文件中写入数据或修改 Excel 样式时，都是由该类来实现的。该类中包含一个 createSheet()的抽象方法，用于创建工作表，语法结构如下：

```
public abstract WritableSheet createSheet(String name, int index);
```

参数说明

❶ name：表示工作表的名称。

❷ index：表示创建的是第几个工作表。

功能：

创建工作表。createSheet()抽象方法的具体实现是在 jxl.write.biff.WritableWorkbookImpl 类中实现的，WritableWorkbookImpl 类是 WritableWorkbook 的派生类。

说明：JXL 组件的 jar 包以及源码文件，可以到 http://www.andykhan.com/jexcelapi/download.html 网站进行下载，下载之后将 jxl.jar 文件直接复制到 Web 工程的 WEB-INF/lib 目录下即可。

设计过程

（1）创建操作 Excel 的工具类 ExcelOperationUtil.java，首先在该类中导入 jxl 包，然后编写创建 Excel 工作簿的方法 createExcelFile()，参数 filePath 表示文件的路径，参数 fileName 表示文件名。其关键代码如下：

```
import jxl.*;
import jxl.write.WritableSheet;
import jxl.write.WritableWorkbook;
public class ExcelOperationUtil {
 public boolean createExcelFile(String filePath,String fileName){
        try {
                WritableWorkbook book = Workbook.createWorkbook(new File(filePath,fileName));      //根据 File 文件对象创建工作簿
                WritableSheet sheet = book.createSheet("第一页", 0);                              //创建工作表，设置名称为“第一页”
                book.write();                                                                     //写入数据
                book.close();                                                                     //关闭文件
                return true;
        } catch (Exception e) {
                System.out.println("异常信息："+e.getMessage());
                e.printStackTrace();
                return false;
        }
 }
}
```

（2）创建 index.jsp 页面，在该页面中获取表单输入的文件路径和文件名，然后调用 ExcelOperationUtil 类的 createExcelFile()方法创建 Excel 文件。index.jsp 页面的关键代码如下：

```
<%
request.setCharacterEncoding("UTF-8");
String filePath = request.getParameter("filePath");                                  //文件路径
String fileName = request.getParameter("fileName");                                  //文件名
if(filePath!=null&&!filePath.equals("")&&fileName!=null&&!fileName.equals("")){     //判断文件路径和文件名是否为空
        if(fileName.lastIndexOf(".")==-1){                                           //判断是否包含文件名后缀
                fileName = fileName+".xls";
        }
        String extendStr = fileName.substring(fileName.lastIndexOf("."));
        if(!extendStr.equals(".xls")){                                               //判断文件名后缀是否为 Excel 文件类型
                out.println("<script>alert('文件错误，文件类型为.xls！');</script>");
        }else{                                                                       //文件为 Excel 文件类型时，调用方法进行创建
                ExcelOperationUtil excelUtil = new ExcelOperationUtil();
                boolean res = excelUtil.createExcelFile(filePath,fileName);
                if(res)
                        out.println("<script>alert('创建成功！');</script>");
        }
}
%>
```

秘笈心法

在获取表单输入的数据之后，一定要对数据的正确性进行合理判断，否则直接将表单数据提交，可能会产生异常，这也是程序开发必不可少的步骤之一。

实例 533	将表单信息导出到 Excel 光盘位置：光盘\MR\21\533	初级 实用指数：★★★

实例说明

本实例将介绍如何应用 JXL 组件，将 JSP 表单数据导出 Excel 工作表中。运行本实例，如图 21.2 所示，输入员工信息，当单击“导出”按钮后，员工信息将被导出 Excel 工作表中。

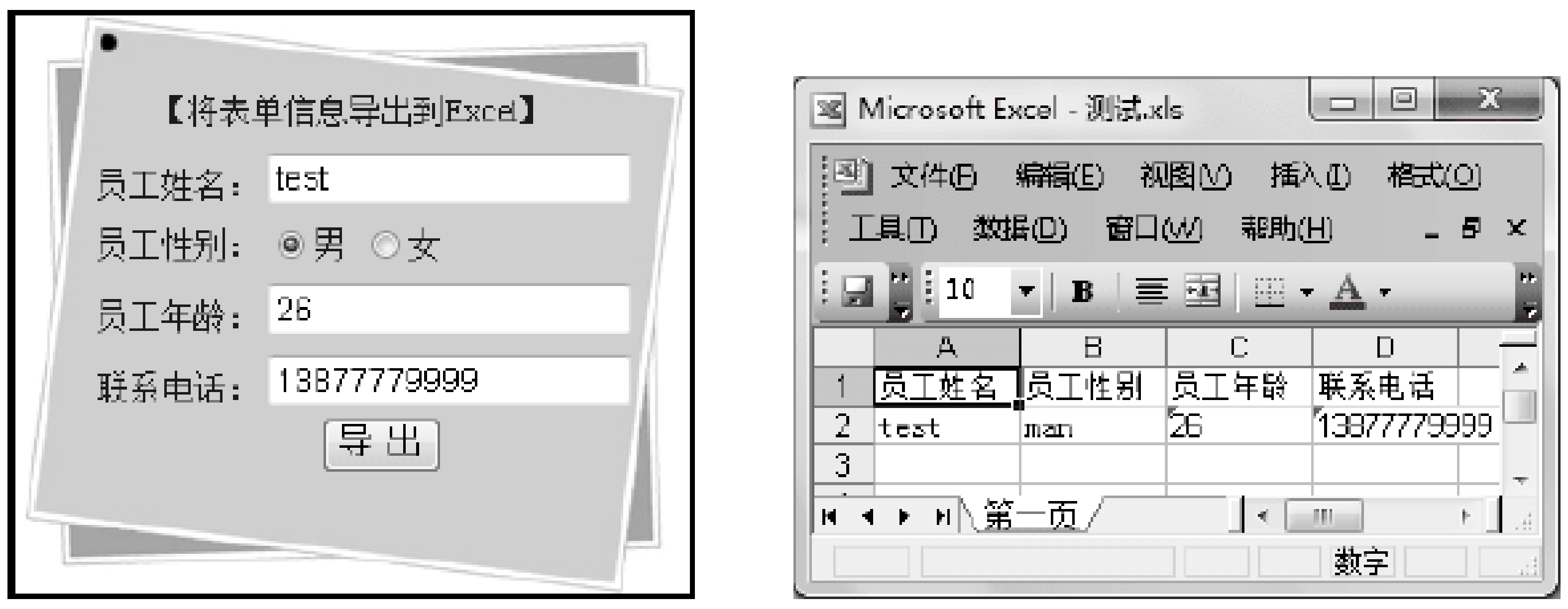


图 21.2　将表单信息添加到 Excel 工作表中

关键技术

本实例主要应用 jxl.write.WritableSheet 接口的 addCell()方法向 Excel 工作表的单元格中添加数据。WritableSheet 对象是由 WritableWorkbook 对象创建的。WritableSheet 是一个接口，该接口的所有方法是在其派生类 jxl.write.biff.WritableSheetImpl 中实现的。addCell()方法的语法结构如下：

```
public void addCell(jxl.write.WritableCell arg0) throws jxl.write.WriteException, jxl.write.biff.RowsExceededException;
```

参数说明

jxl.write.WritableCell：表示一个单元格对象。在调用 addCell()方法之前，首先需要创建单元格对象 WritableCell。

WritableCell 也是一个接口，具体实现是由它的子类完成的。本实例创建单元格对象时，用的是 WritableCell 接口的 Label 类。Label 类的构造方法如下：

```
Label label = new Label(int col,int row,String cont)
```

参数说明

❶ col：表示 Excel 工作表的单元格所在列的索引。索引从 0 开始。

❷ row：表示 Excel 工作表的单元格所在行的索引。索引从 0 开始。

❸ cont：表示添加到 Excel 单元格中的字符串内容。

设计过程

（1）创建用于封装表单信息的 JavaBean 类 Employee，其关键代码如下：

```
public class Employee {
        private String name;
        private String sex;
        private int age;
        private String telephone;
……//此处省略了属性的 getXXX()方法和 setXXX()方法
}
```

（2）创建用于操作 Excel 的工具类 ExcelOperationUtil，在该类中编写用于向 Excel 工作表中添加信息的方法 addInfoToExcel()，其关键代码如下：

```
public boolean addInfoToExcel(Employee emp){
        try {
                WritableWorkbook book = Workbook.createWorkbook(new File("E:\\test\\测试.xls"));    //创建工作簿
                WritableSheet sheet = book.createSheet("第一页", 0);                                  //创建工作表
                Label name = new Label(0, 0, "员工姓名");                                            //创建单元格
                Label sex = new Label(1,0,"员工性别");                                               //创建单元格
                Label age = new Label(2,0,"员工年龄");                                               //创建单元格
                Label telephone = new Label(3,0,"联系电话");                                         //创建单元格
                Label name_value = new Label(0,1,emp.getName());                                    //创建单元格
                Label sex_value = new Label(1,1,emp.getSex());                                      //创建单元格
                Label age_value = new Label(2,1,emp.getAge()+"");                                   //创建单元格
                Label tele_value = new Label(3,1,emp.getTelephone());                               //创建单元格
                sheet.addCell(name);                                                                 //向工作表单元格中添加内容
                sheet.addCell(sex);                                                                  //向工作表单元格中添加内容
                sheet.addCell(age);                                                                  //向工作表单元格中添加内容
                sheet.addCell(telephone);                                                            //向工作表单元格中添加内容
                sheet.addCell(name_value);                                                           //向工作表单元格中添加内容
```

```
            sheet.addCell(sex_value);                                   //向工作表单元格中添加内容
            sheet.addCell(age_value);                                   //向工作表单元格中添加内容
            sheet.addCell(tele_value);                                  //向工作表单元格中添加内容
            book.write();                                               //将所有内容写入 Excel
            book.close();                                               //关闭
            return true;
        } catch (Exception e) {
            System.out.println("异常信息："+e.getMessage());
            e.printStackTrace();
            return false;
        }
}
```

（3）创建 index.jsp 页面，获取用户输入的表单信息并封装在 Employee 对象中，然后调用 ExcelOperationUtil 类的 addInfoToExcel()方法，将表单信息导出 Excel 工作表中，其关键代码如下：

```
<%
String submit = request.getParameter("submit");
if(submit!=null){
        String name = new String(request.getParameter("name"));
        String    sex= new String( request.getParameter("sex"));
        String age = request.getParameter("age");
        String telephone = request.getParameter("telephone");
        Employee emp = new Employee();
        emp.setName(name);
        emp.setSex(sex);
        emp.setAge(Integer.parseInt(age));
        emp.setTelephone(telephone);
        ExcelOperationUtil excelUtil = new ExcelOperationUtil();
        boolean res = excelUtil.addInfoToExcel(emp);
        if(res)
                out.println("<script>alert('信息导出成功！');</script>");
}
 %>
```

秘笈心法

在 Excel 单元格中添加完信息之后，千万不要忘记调用 Excel 工作簿 WritableWorkbook 对象的 write()方法执行写入操作，因为在调用 write()方法之前，数据是存储在缓存中的，调用 write()方法的目的是将缓存中的数据写到文件中。

实例 534	向 Excel 工作表中添加数值 光盘位置：光盘\MR\21\534	初级 实用指数：★★★

实例说明

实例 533 介绍了如何向 Excel 工作表中添加数据，其所有信息都是以字符串的格式添加的。本实例将介绍如何应用 JXL 组件，将数值信息添加到 Excel 工作表中。运行本实例，如图 21.3 所示，输入商品名称、商品数量以及商品单价，当单击“导出”按钮后，商品信息将被添加到 Excel 工作表中，其中商品数量和商品单价是以数值的格式添加的。

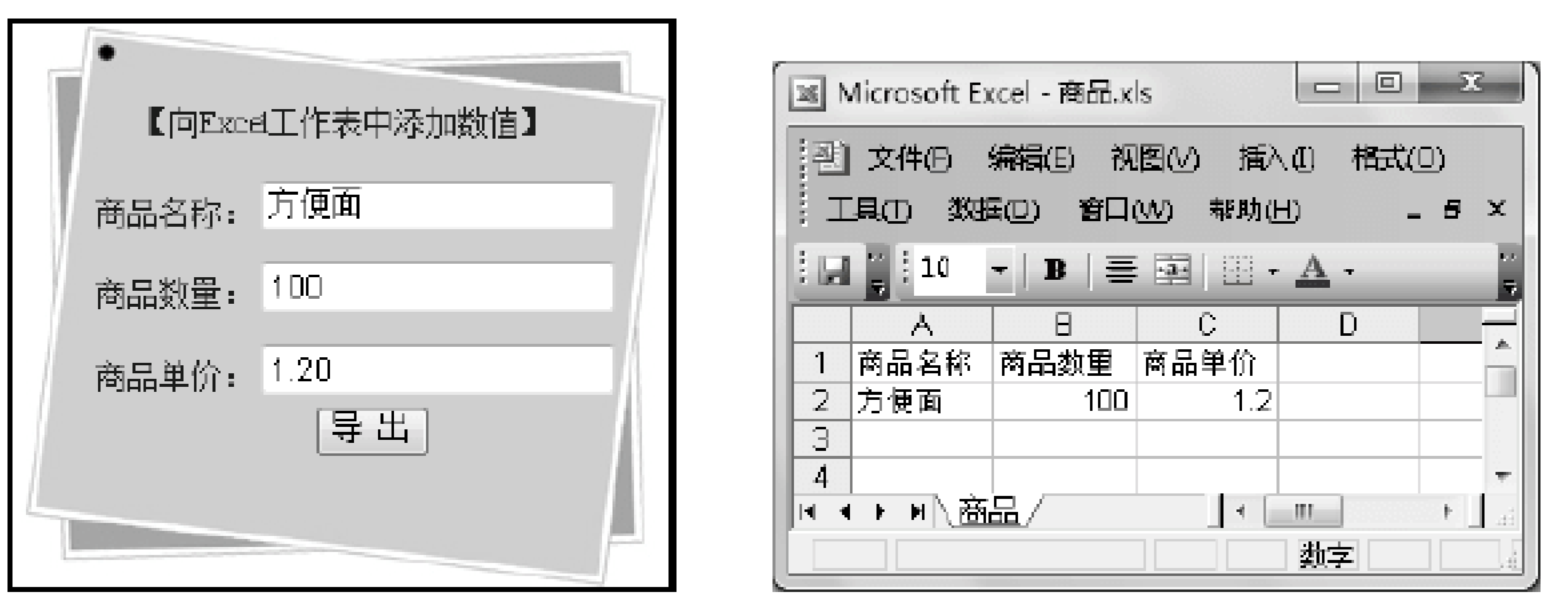


图 21.3　向 Excel 工作表中添加数值

关键技术

向 Excel 工作表的单元格中添加数值类型的数据，主要是应用 jxl.write.Number 类。该类是 WritableCell 抽象接口的实现类。因此，创建的 Number 对象也是一个单元格对象，也可以作为工作表 WritableSheet 对象的 addCell()方法的参数直接添加。Number 类的构造方法语法如下：

```
public Number(int c, int r, double val)
```

参数说明

❶ c：表示 Excel 工作表的单元格所在列的索引。索引从 0 开始。

❷ r：表示 Excel 工作表的单元格所在行的索引。索引从 0 开始。

❸ val：表示添加到 Excel 单元格中的 double 类型的数值。

设计过程

（1）创建用于操作 Excel 的工具类 ExcelOperationUtil，在该类中编写用于向 Excel 工作表中添加数值信息的方法 addInfoToExcel()，其关键代码如下：

```
public boolean addInfoToExcel(String name,String count,String price){
try {
        WritableWorkbook book = Workbook.createWorkbook(new File("E:\\test\\商品.xls"));
        WritableSheet sheet = book.createSheet("商品", 0);
        Label label_name = new Label(0,0,"商品名称");
        Label label_count = new Label(1,0,"商品数量");
        Label label_price = new Label(2,0,"商品单价");
        Label name_value = new Label(0,1,name);
        //创建数值类型的单元格对象
        jxl.write.Number count_value = new jxl.write.Number(1,1,Double.parseDouble(count));
        //创建数值类型的单元格对象
        jxl.write.Number price_value = new jxl.write.Number(2,1,Double.parseDouble(price));
        sheet.addCell(label_name);
        sheet.addCell(label_count);
        sheet.addCell(label_price);
        sheet.addCell(name_value);
        sheet.addCell(price_value);
        sheet.addCell(count_value);
        book.write();
        book.close();
        return true;
} catch (Exception e) {
        System.out.println("异常信息："+e.getMessage());
        e.printStackTrace();
        return false;
}
}
```

（2）创建 save.jsp 页面，获取用户输入的表单信息，然后调用 ExcelOperationUtil 类的 addInfoToExcel()方法，将包含数值的表单信息导出 Excel 工作表中，其关键代码如下：

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8" contentType="text/html; charset=UTF-8"%>
<%@ page import="com.lh.util.ExcelOperationUtil"   %>
<%
String submit = request.getParameter("submit");
if(submit!=null){
        String name = new String(request.getParameter("name").getBytes("ISO-8859-1"),"UTF-8");
        String   count= new String( request.getParameter("count"));
        String price = request.getParameter("price");
        ExcelOperationUtil excelUtil = new ExcelOperationUtil();
        boolean res = excelUtil.addInfoToExcel(name,count,price);
        if(res)
                out.println("<script>alert('信息导出成功！');window.location.href = 'index.jsp';</script>");
}
 %>
```

秘笈心法

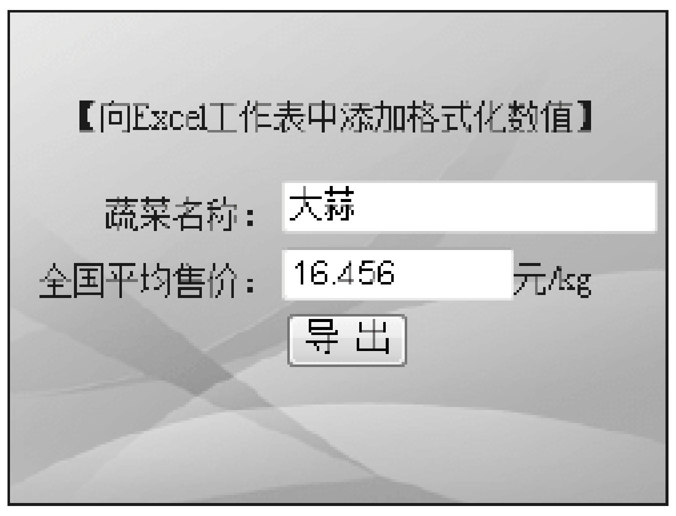
在实例应用中，由于需要根据数值类型的数据进行汇总，因此需要应用 jxl.write.Number 对象向 Excel 工作表中添加数值类型的数据。

实例 535　向 Excel 工作表中添加格式化数值　初级

光盘位置：光盘\MR\21\535　实用指数：★★★

实例说明

本实例将介绍如何应用 JXL 组件，将格式化的数值信息添加到 Excel 工作表中。运行本实例，如图 21.4 所示，输入蔬菜名称和全国平均售价，当单击“导出”按钮后，蔬菜信息将被添加到 Excel 工作表中，其中蔬菜的全国平均售价值的小数点后保留两位。



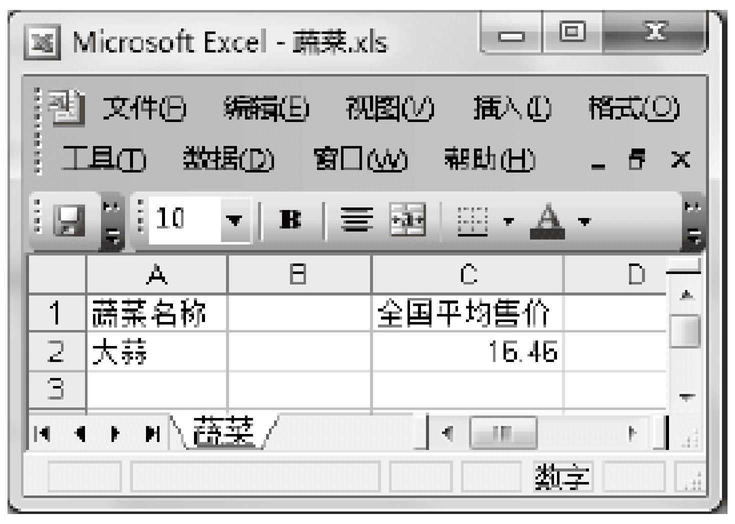


图 21.4　向 Excel 工作表中添加格式化数值

关键技术

向 Excel 工作表中添加格式化数值，主要应用的是 jxl.write.NumberFormat 类和 jxl.write.WritableCellFormat 类。下面对这两个类进行详细介绍。

（1）NumberFormat 类

该类表示数值格式化模板，构造方法如下：

```
public NumberFormat(String format)
```

参数说明

format：表示格式化字符串。通常用“#”表示格式化字符串，如设置数值的小数点后保留两位，那么格式化字符串为“#.##”。

（2）WritableCellFormat 类

该类表示单元格的格式化模板，创建该模板对象后，可以设置单元格的样式以及单元格内容的样式。该类的其中一个构造方法如下：

```
public WritableCellFormat(DisplayFormat format)
```

参数说明

format：该参数为 jxl.biff.DisplayFormat 接口类型，而 NumberFormat 为这个接口的其中一个实现类，所以可以将 NumberFormat 对象作为 WritableCellFormat 类的构造方法参数。

设计过程

创建用于操作 Excel 的工具类 ExcelOperationUtil，在该类中编写用于向 Excel 工作表中添加格式化数值信息的方法 addInfoToExcel()，其关键代码如下：

```
public boolean addInfoToExcel(String name,String price){
try {
        WritableWorkbook book = Workbook.createWorkbook(new File("E:\\test\\蔬菜.xls"));
        WritableSheet sheet = book.createSheet("蔬菜", 0);
        Label label_name = new Label(0,0,"蔬菜名称");
        Label label_price = new Label(1,0,"全国平均售价");
```

```
        Label name_value = new Label(0,1,name);
        NumberFormat format = new NumberFormat("#.##");//数值格式化模板
        WritableCellFormat cellFormat = new WritableCellFormat(format);
        //创建数值类型的单元格对象，并且应用指定格式
        jxl.write.Number price_value = new jxl.write.Number(1,1,Double.parseDouble(price),cellFormat);
        sheet.addCell(label_name);
        sheet.addCell(label_price);
        sheet.addCell(name_value);
        sheet.addCell(price_value);
        book.write();
        book.close();
        return true;
} catch (Exception e) {
        System.out.println("异常信息："+e.getMessage());
        e.printStackTrace();
        return false;
}
}
```

秘笈心法

在实际应用中，有时需要对 Excel 工作表中添加的数值信息进行格式化，如要求数值的小数点后只保留两位有效数字。

实例 536 向 Excel 工作表中添加 boolean 值

光盘位置：光盘\MR\21\536

初级

实用指数：★★★

实例说明

本实例将介绍如何应用 JXL 组件，将表示 true 或 false 的布尔值添加到 Excel 工作表中。运行本实例，如图 21.5 所示，输入蔬菜名称和市场平均售价，然后选择是否涨价，当单击“导出”按钮后，蔬菜信息将被添加到 Excel 工作表中，其中预测是否涨价的值为 TRUE 或 FALSE。



图 21.5 向 Excel 工作表中添加 boolean 值

关键技术

向 Excel 工作表中添加 boolean 值，主要应用的是 jxl.write.Boolean 类，该类实现了单元格抽象接口 WritableCell，创建该类的对象，表示一个 boolean 值的单元格对象。jxl.write.Boolean 类的构造方法如下：

```
public Boolean(int col, int row, boolean val)
```

参数说明

❶ col：表示 Excel 工作表的单元格所在列的索引。索引从 0 开始。

❷ row：表示 Excel 工作表的单元格所在行的索引。索引从 0 开始。

❸ val：表示添加到 Excel 单元格中的 boolean 类型的值。

设计过程

创建用于操作 Excel 的工具类 ExcelOperationUtil，在该类中编写用于向 Excel 工作表中添加 boolean 值的方

法 addInfoToExcel()，其关键代码如下：

```
public boolean addInfoToExcel(String name,String price,String isUp){
 try {
        WritableWorkbook book = Workbook.createWorkbook(new File("E:\\test\\蔬菜.xls"));
        WritableSheet sheet = book.createSheet("蔬菜", 0);
        Label label_name = new Label(0,0,"蔬菜名称");
        Label label_price = new Label(1,0,"市场平均售价");
        Label label_is = new Label(2,0,"预测是否涨价");
        Label name_value = new Label(0,1,name);
        jxl.write.Boolean b;
        if(isUp.equals("yes")){                        //根据用户选择的值，创建 jxl.write.Boolean 对象
                b = new jxl.write.Boolean(2,1,true);
        }else{                                          //根据用户选择的值，创建 jxl.write.Boolean 对象
                b = new jxl.write.Boolean(2,1,false);
        }
        NumberFormat format = new NumberFormat("#.##");  //数值格式化模板
        WritableCellFormat cellFormat = new WritableCellFormat(format);
        //创建数值类型的单元格对象，并且应用指定格式
        jxl.write.Number price_value = new jxl.write.Number(1,1,Double.parseDouble(price),cellFormat);
        sheet.addCell(label_name);
        sheet.addCell(label_price);
        sheet.addCell(name_value);
        sheet.addCell(price_value);
        sheet.addCell(label_is);
        sheet.addCell(b);                               //将 boolean 值添加到指定的单元格
        book.write();
        book.close();
        return true;
 } catch (Exception e) {
        System.out.println("异常信息："+e.getMessage());
        e.printStackTrace();
        return false;
 }
}
```

秘笈心法

在实例应用中，有时需要向 Excel 工作表的某一列中添加 boolean 值，此时就需要应用 jxl.write.Boolean 类进行处理。

实例 537　向 Excel 工作表中添加日期时间

光盘位置：光盘\MR\21\537

初级

实用指数：★★★

实例说明

本实例将介绍如何应用 JXL 组件，向 Excel 工作表中添加日期时间类型的数据。运行本实例，如图 21.6 所示，输入用户信息，当单击“导出”按钮后，用户信息将被添加到 Excel 工作表中，其中生日信息为日期时间类型。

关键技术

向 Excel 工作表中添加日期时间值，主要应用的是 jxl.write.DateTime 类，该类实现了单元格抽象接口 WritableCell，创建该类的对象，表示一个日期时间类型的单元格对象。jxl.write.DateTime 类的构造方法如下：

```
public DateTime(int col, int row, java.util.Date date)
```

参数说明

❶ col：表示 Excel 工作表的单元格所在列的索引。索引从 0 开始。

❷ row：表示 Excel 工作表的单元格所在行的索引。索引从 0 开始。

❸ date：java.util.Date 类型的参数。

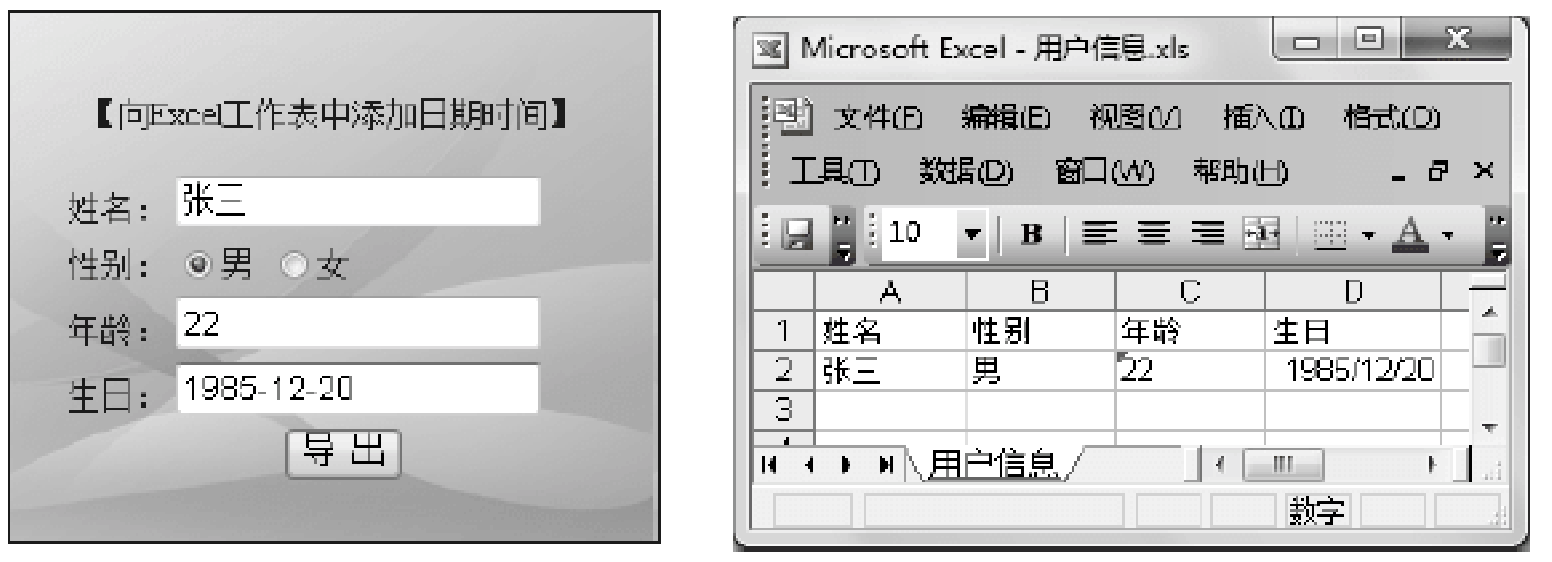


图 21.6　向 Excel 工作表中添加日期时间

设计过程

创建用于操作 Excel 的工具类 ExcelOperationUtil，在该类中编写用于向 Excel 工作表中添加用户信息的方法 addInfoToExcel()，其中添加的用户生日为日期时间类型，其关键代码如下：

```
public boolean addInfoToExcel(String name,String sex,String age,String birthday){
 try {
        WritableWorkbook book = Workbook.createWorkbook(new File("E:\\test\\用户信息.xls"));
        WritableSheet sheet = book.createSheet("用户信息", 0);
        Label label_name = new Label(0,0,"姓名");
        Label label_sex = new Label(1,0,"性别");
        Label label_age = new Label(2,0,"年龄");
        Label label_birthday = new Label(3,0,"生日");
        Label name_value = new Label(0,1,name);
        Label sex_value = new Label(1,1,sex);
        Label age_value = new Label(2,1,age);
        String[] dateStr = birthday.split("-");                    //分割生日字符串
        Calendar c = Calendar.getInstance();                       //创建 Calendar 对象
        //将用户生日的年月日设置为 Calendar 对象的年月日
        c.set(Integer.parseInt(dateStr[0]), (Integer.parseInt(dateStr[1])-1), Integer.parseInt(dateStr[2]));
        //创建日期时间类型的单元格对象，并将表示用户生日的 Calendar 对象添加到该对象中
        jxl.write.DateTime date_value = new jxl.write.DateTime(3,1,c.getTime());
        sheet.addCell(label_name);
        sheet.addCell(label_sex);
        sheet.addCell(label_age);
        sheet.addCell(label_birthday);
        sheet.addCell(name_value);
        sheet.addCell(sex_value);
        sheet.addCell(age_value);
        sheet.addCell(date_value);
        book.write();
        book.close();
        return true;
 } catch (Exception e) {
        System.out.println("异常信息："+e.getMessage());
        e.printStackTrace();
        return false;
 }
}
```

秘笈心法

在实例应用中，经常需要向 Excel 工作表中添加日期时间类型的数据，因此需要应用 jxl.write.DateTime 类进行处理。需要注意的是，Calendar 对象表示的月份范围在 0～11 之间，所以在为 Calendar 对象的月份设置值时，必须将月份值减 1。

实例 538	向 Excel 工作表中添加格式化日期时间 光盘位置：光盘\MR\21\538	初级 实用指数：★★★

实例说明

实例 537 已经介绍了如何向 Excel 工作表中添加日期时间，其格式为默认类型。有时根据实际需求，需要

添加不同格式的日期时间值。本实例将介绍如何应用 JXL 组件，向 Excel 工作表中添加格式化的日期时间类型的数据。运行本实例，如图 21.7 所示，输入用户信息，当单击“导出”按钮后，用户信息将被添加到 Excel 工作表中，其中生日类型为 yyyy-mm-dd。

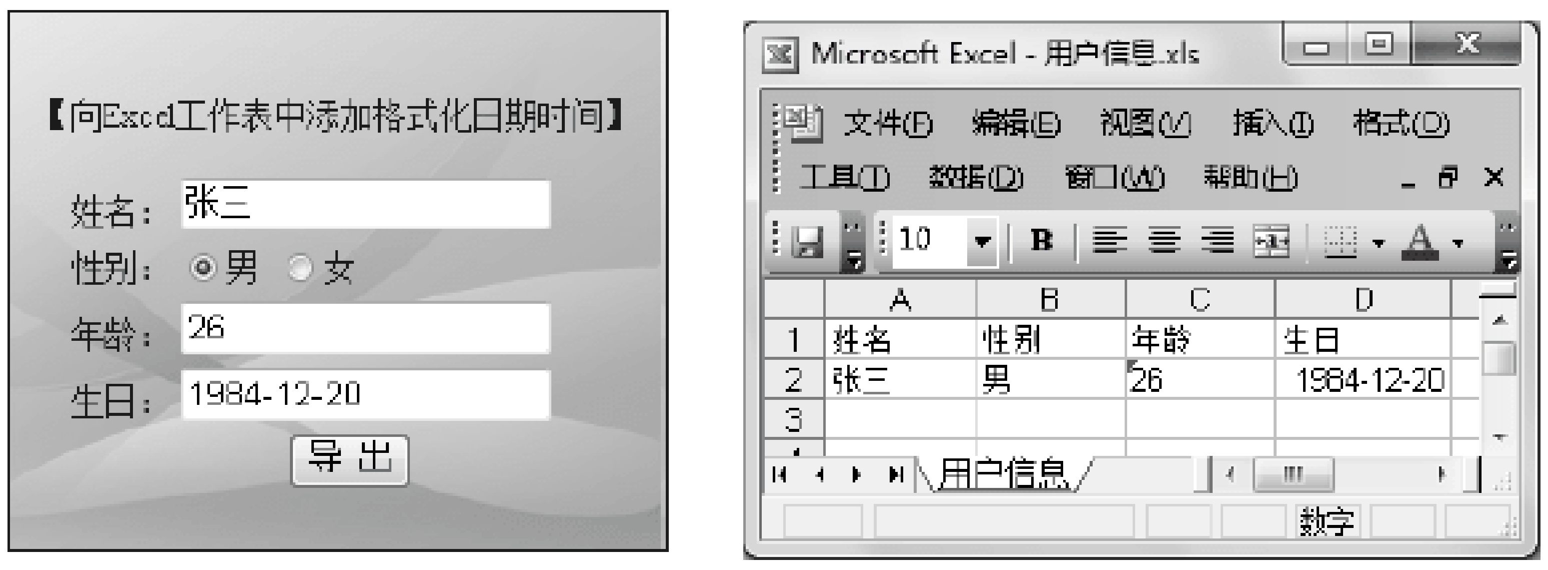


图 21.7　向 Excel 工作表中添加格式化日期时间

关键技术

向 Excel 工作表中添加格式化的日期时间值，主要应用的是 jxl.write.DateFormat 类，该类表示日期时间格式器。jxl.write.DateFormat 类的构造方法如下：

```
public DateFormat(java.lang.String format)
```

参数说明

format：表示日期时间的格式化字符串。如 yyyy-mm-dd、yyyy mm dd、dd mm yyyy、yyyy MM dd hh:mm:ss 等。

设计过程

创建用于操作 Excel 的工具类 ExcelOperationUtil，在该类中编写用于向 Excel 工作表中添加用户信息的方法 addInfoToExcel()，其中添加的用户生日格式为 yyyy-mm-dd，其关键代码如下：

```
public boolean addInfoToExcel(String name,String sex,String age,String birthday){
 try {
        WritableWorkbook book = Workbook.createWorkbook(new File("E:\\test\\用户信息.xls"));
        WritableSheet sheet = book.createSheet("用户信息", 0);
        Label label_name = new Label(0,0,"姓名");
        Label label_sex = new Label(1,0,"性别");
        Label label_age = new Label(2,0,"年龄");
        Label label_birthday = new Label(3,0,"生日");
        Label name_value = new Label(0,1,name);
        Label sex_value = new Label(1,1,sex);
        Label age_value = new Label(2,1,age);
        String[] dateStr = birthday.split("-");        //分割生日字符串
        Calendar c = Calendar.getInstance();           //创建 Calendar 对象
        //将用户生日的年月日设置为 Calendar 对象的年月日
        c.set(Integer.parseInt(dateStr[0]), (Integer.parseInt(dateStr[1])-1), Integer.parseInt(dateStr[2]));
        jxl.write.DateFormat format = new jxl.write.DateFormat("yyyy-mm-dd");
        jxl.write.WritableCellFormat cellFormat = new jxl.write.WritableCellFormat(format);
        //创建日期时间类型的单元格对象，并将表示用户生日的 Calendar 对象添加到该对象中
        jxl.write.DateTime date_value = new jxl.write.DateTime(3,1,c.getTime(),cellFormat);
        sheet.addCell(label_name);
        sheet.addCell(label_sex);
        sheet.addCell(label_age);
        sheet.addCell(label_birthday);
        sheet.addCell(name_value);
        sheet.addCell(sex_value);
        sheet.addCell(age_value);
        sheet.addCell(date_value);
        book.write();
        book.close();
        return true;
 } catch (Exception e) {
        System.out.println("异常信息："+e.getMessage());
        e.printStackTrace();
        return false;
 }
}
```

秘笈心法

在创建 jxl.write.DateFormat 对象时，其构造方法的格式化参数中表示年份的字符串必须为小写英文字母 yyyy，如果是大写的 YYYY，将抛出异常。

实例 539 设置 Excel 工作表字体样式

光盘位置：光盘\MR\21\539

初级

实用指数：★★★

实例说明

本实例将介绍如何应用 JXL 组件设置 Excel 工作表字体样式。运行本实例，如图 21.8 所示，输入用户信息，当单击“导出”按钮后，用户信息将被添加到 Excel 工作表中，并且设置了用户姓名的字体样式。

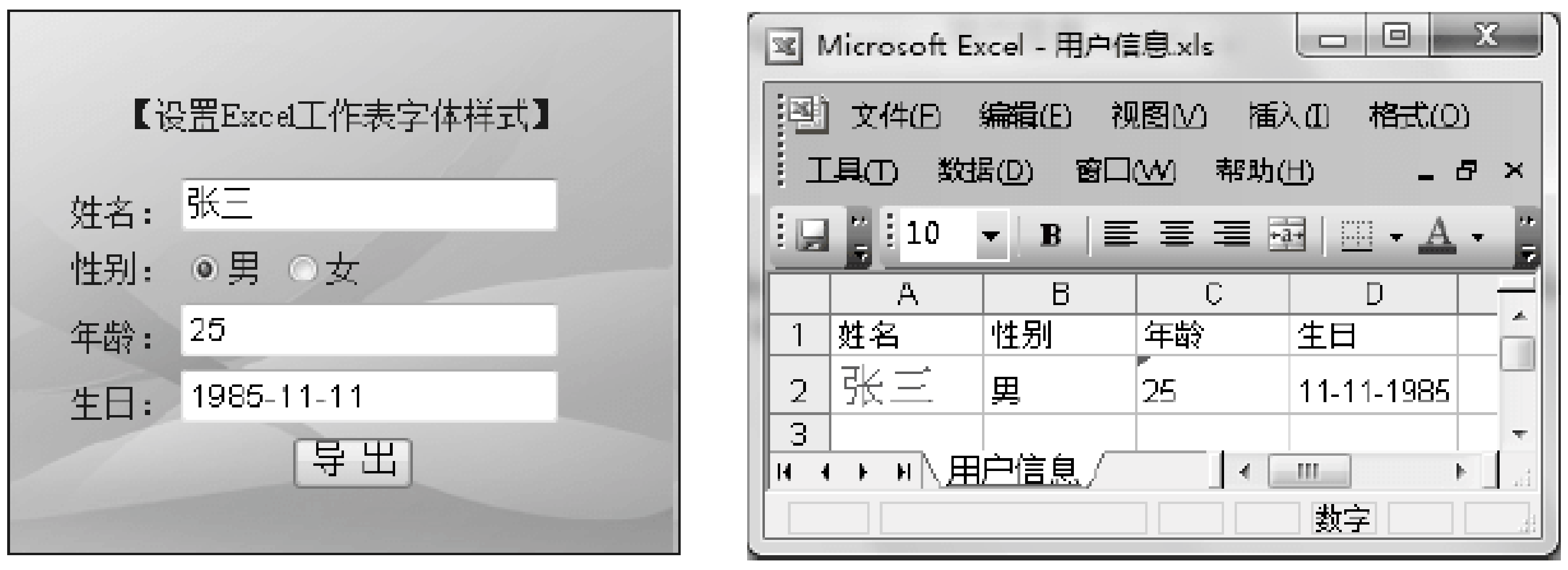


图 21.8　设置 Excel 工作表字体样式

关键技术

设置 Excel 工作表的字体样式，主要应用的是 jxl.write.WritableFont 类，该类主要用于设置字体的样式，如字体大小、字体颜色、是否带有下划线等。jxl.write.WritableFont 类的构造方法如下：

```
public WritableFont(WritableFont.FontName fn, int ps, WritableFont.BoldStyle bs, boolean it, jxl.format.UnderlineStyle us, jxl.format.Colour c)
```

参数说明

❶ fn：设置字体的样式名称。例如 TIMES（Times New Roman）、ARIAL（Arial）、COURIER（Courier New）、TAHOMA（Tahoma）等。

❷ ps：设置字体的大小。

❸ bs：设置字体是否加粗。取值为 NO_BOLD（不加粗）和 BOLD（加粗）。

❹ it：设置字体是否倾斜。取值为 true（倾斜）和 false（不倾斜）。

❺ us：设置字体是否带有下划线。取值包括 NO_UNDERLINE（无下划线）、SINGLE（单下划线）、SINGLE_ACCOUNTING（会计用单下划线）、DOUBLE（双下划线）和 DOUBLE_ACCOUNTING（会计用双下划线）。

❻ c：设置字体颜色。包括一些常用的字体颜色，如 RED、BLACK、GREEN、BLUE、PINK、YELLOW、GRAY 等。

设计过程

创建用于操作 Excel 的工具类 ExcelOperationUtil，在该类中编写用于向 Excel 工作表中添加用户信息的方法 addInfoToExcel()，在该方法中设置了用户姓名单元格的字体样式，其关键代码如下：

```
public boolean addInfoToExcel(String name,String sex,String age,String birthday){
 try {
        WritableWorkbook book = Workbook.createWorkbook(new File("E:\\test\\用户信息.xls"));
        WritableSheet sheet = book.createSheet("用户信息", 0);
        Label label_name = new Label(0,0,"姓名");
        Label label_sex = new Label(1,0,"性别");
        Label label_age = new Label(2,0,"年龄");
```

```
        Label label_birthday = new Label(3,0,"生日");
        jxl.write.WritableFont font = new jxl.write.WritableFont(WritableFont.ARIAL, 15, WritableFont.BOLD, false,
UnderlineStyle.SINGLE_ACCOUNTING, jxl.format.Colour.GREEN);                    //设置字体样式
        jxl.write.WritableCellFormat fontFormat = new jxl.write.WritableCellFormat(font);//创建单元格的格式器对象，并应用字体样式
        Label name_value = new Label(0,1,name,fontFormat);                        //将字体样式应用于此单元格
        Label sex_value = new Label(1,1,sex);
        Label age_value = new Label(2,1,age);
        Label birthday_value = new Label(3,1,birthday);
        sheet.addCell(label_name);
        sheet.addCell(label_sex);
        sheet.addCell(label_age);
        sheet.addCell(label_birthday);
        sheet.addCell(name_value);
        sheet.addCell(sex_value);
        sheet.addCell(age_value);
        sheet.addCell(birthday_value);
        book.write();
        book.close();
        return true;
} catch (Exception e) {
        System.out.println("异常信息："+e.getMessage());
        e.printStackTrace();
        return false;
}
}
```

秘笈心法

在实际开发应用过程中，经常需要设置 Excel 工作表的字体样式，此时就需要应用 jxl.write.WritableFont 类进行处理，因此读者应该掌握这个类的用法。

实例 540　合并 Excel 工作表的单元格

初级

光盘位置：光盘\MR\21\540

实用指数：★★★

实例说明

本实例将介绍如何应用 JXL 组件，对 Excel 工作表的单元格进行合并。运行本实例，如图 21.9 所示，输入用户信息，当单击“导出”按钮后，用户信息将被添加到 Excel 工作表中，并且对单元格进行了合并操作。

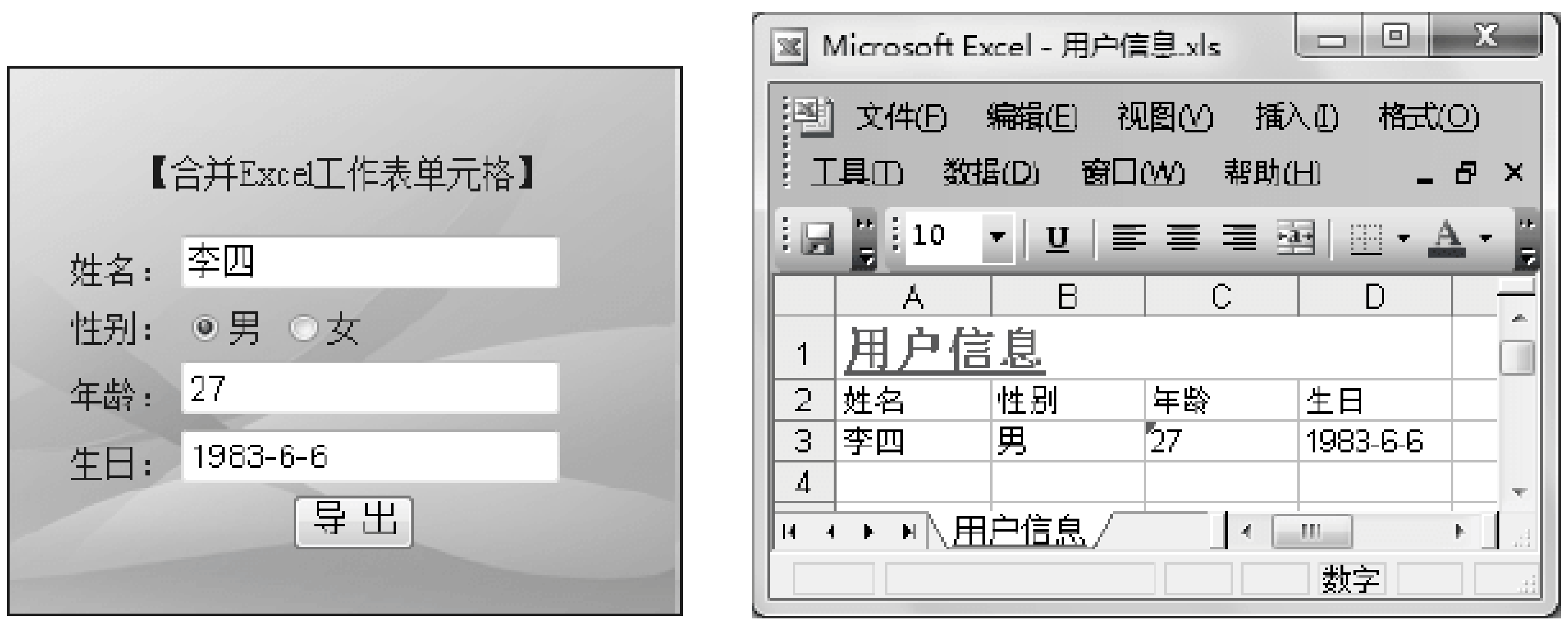


图 21.9　合并 Excel 工作表的单元格

关键技术

对 Excel 工作表的单元格进行合并，主要应用的是 jxl.write.WritableSheet 接口的 mergeCells()方法，该方法的语法结构如下：

```
public jxl.Range mergeCells(int col1, int row1, int col2, int row2) throws jxl.write.WriteException, jxl.write.biff.RowsExceededException;
```

参数说明

❶ col1：表示要合并单元格的起始位置单元格的列索引。索引从 0 开始。

❷ row1：表示要合并单元格的起始位置单元格的行索引。索引从 0 开始。

❸ col2：表示要合并单元格的结束位置单元格的列索引。索引从 0 开始。

❹ row2：表示要合并单元格的结束位置单元格的行索引。索引从 0 开始。

设计过程

创建用于操作 Excel 的工具类 ExcelOperationUtil，在该类中编写用于向 Excel 工作表中添加用户信息的方法 addInfoToExcel()，在该方法中对 Excel 工作表的单元格进行合并操作，其关键代码如下：

```
public boolean addInfoToExcel(String name,String sex,String age,String birthday){
try {
        WritableWorkbook book = Workbook.createWorkbook(new File("E:\\test\\用户信息.xls"));
        WritableSheet sheet = book.createSheet("用户信息", 0);
        jxl.write.WritableFont font = new jxl.write.WritableFont(WritableFont.ARIAL, 15, WritableFont.BOLD, false,UnderlineStyle.SINGLE, jxl.format.
Colour.GREEN);                                                    //设置字体样式
        jxl.write.WritableCellFormat fontFormat = new jxl.write.WritableCellFormat(font);
        Label label_title = new Label(0,0,"用户信息",fontFormat);    //单元格应用字体样式
        sheet.mergeCells(0,0,3,0);                                //合并第 1 行的第 1～第 3 个单元格
        Label label_name = new Label(0,1,"姓名");
        Label label_sex = new Label(1,1,"性别");
        Label label_age = new Label(2,1,"年龄");
        Label label_birthday = new Label(3,1,"生日");
        Label name_value = new Label(0,2,name);
        Label sex_value = new Label(1,2,sex);
        Label age_value = new Label(2,2,age);
        Label birthday_value = new Label(3,2,birthday);
        sheet.addCell(label_title);
        sheet.addCell(label_name);
        sheet.addCell(label_sex);
        sheet.addCell(label_age);
        sheet.addCell(label_birthday);
        sheet.addCell(name_value);
        sheet.addCell(sex_value);
        sheet.addCell(age_value);
        sheet.addCell(birthday_value);
        book.write();
        book.close();
        return true;
} catch (Exception e) {
        System.out.println("异常信息："+e.getMessage());
        e.printStackTrace();
        return false;
}
}
```

秘笈心法

应用 jxl.write.WritableSheet 类的 mergeCells()方法合并单元格既可以是横向的，也可以是纵向的。需要注意的是，合并后的单元格不能再次进行合并，否则会触发异常。

实例 541	设置 Excel 工作表的单元格内容水平居中 光盘位置：光盘\MR\21\541	初级 实用指数：★★★

实例说明

在实例 540 中实现了合并单元格的操作，但是在合并后的单元格中添加的内容默认是左对齐的。本实例将介绍如何应用 JXL 组件，设置 Excel 工作表的单元格内容水平居中。运行本实例，如图 21.10 所示，输入用户信息，当单击“导出”按钮后，用户信息将被添加到 Excel 工作表中，并且设置“用户信息”标题为水平居中显示。



图 21.10　设置 Excel 工作表的单元格内容水平居中

关键技术

设置 Excel 工作表单元格内容水平居中，主要应用的是 jxl.write.WritableCellFormat 类的 setAlignment()方法。WritableCellFormat 用于格式化单元格的样式，如单元格的字体样式、单元格边框样式、背景颜色、单元格凹陷、单元格锁定、单元格内容水平居中以及单元格内容垂直居中等。setAlignment()方法的语法结构如下：

```
public void setAlignment(jxl.format.Alignment alignment) throws jxl.write.WriteException
```

参数说明

alignment：参数类型为 jxl.format.Alignment。Alignment 专门用于设置单元格内容的水平对齐方式，对齐方式的参数为静态参数，因此可以直接用 Alignment 类名进行调用，如 Alignment.LEFT（左对齐）、Align.CENTER（居中对齐）、Alignment.RIGHT（右对齐）等。

设计过程

创建用于操作 Excel 的工具类 ExcelOperationUtil，在该类中编写用于向 Excel 工作表中添加用户信息的方法 addInfoToExcel()，在该方法中设置 Excel 工作表的“用户信息”标题水平居中显示，其关键代码如下：

```
public boolean addInfoToExcel(String name,String sex,String age,String birthday){
 try {
        WritableWorkbook book = Workbook.createWorkbook(new File("E:\\test\\用户信息.xls"));
        WritableSheet sheet = book.createSheet("用户信息", 0);
        jxl.write.WritableFont font = new jxl.write.WritableFont(WritableFont.ARIAL,15, WritableFont.BOLD,false,UnderlineStyle.SINGLE, jxl.format.
Colour.GREEN);                                          //设置字体样式
        jxl.write.WritableCellFormat fontFormat = new jxl.write.WritableCellFormat(font);
        fontFormat.setAlignment(Alignment.CENTRE);      //设置单元格内容水平居中
        Label label_title = new Label(0,0,"用户信息",fontFormat);     //单元格应用字体样式
        sheet.mergeCells(0,0,3,0);                      //合并第 1 行的第 1～第 3 个单元格
        Label label_name = new Label(0,1,"姓名");
        Label label_sex = new Label(1,1,"性别");
        Label label_age = new Label(2,1,"年龄");
        Label label_birthday = new Label(3,1,"生日");
        Label name_value = new Label(0,2,name);
        Label sex_value = new Label(1,2,sex);
        Label age_value = new Label(2,2,age);
        Label birthday_value = new Label(3,2,birthday);
        sheet.addCell(label_title);
        sheet.addCell(label_name);
        sheet.addCell(label_sex);
        sheet.addCell(label_age);
        sheet.addCell(label_birthday);
        sheet.addCell(name_value);
        sheet.addCell(sex_value);
        sheet.addCell(age_value);
        sheet.addCell(birthday_value);
        book.write();
        book.close();
        return true;
 } catch (Exception e) {
        System.out.println("异常信息："+e.getMessage());
        e.printStackTrace();
        return false;
 }
}
```

秘笈心法

jxl.write.WritableCellFormat 类非常重要，通过它可以指定单元格的各种属性。在应用 Excel 电子表格时，对 Excel 工作表单元格的样式设置是必不可少的，所以应该掌握 WritableCellFormat 类的用法。

实例 542	设置 Excel 工作表的行高 光盘位置：光盘\MR\21\542	初级 实用指数：★★★

实例说明

本实例将介绍如何应用 JXL 组件，设置 Excel 工作表的行高度。运行本实例，如图 21.11 所示，输入员工信息，当单击“导出”按钮后，员工信息将被添加到 Excel 工作表中，并且设置“员工信息”标题行的行高为 20 像素。在 Excel 工作表中，使用鼠标单击行的索引，然后单击鼠标右键，在弹出的快捷菜单中选择“行高”命令，即可查看该行的高度。

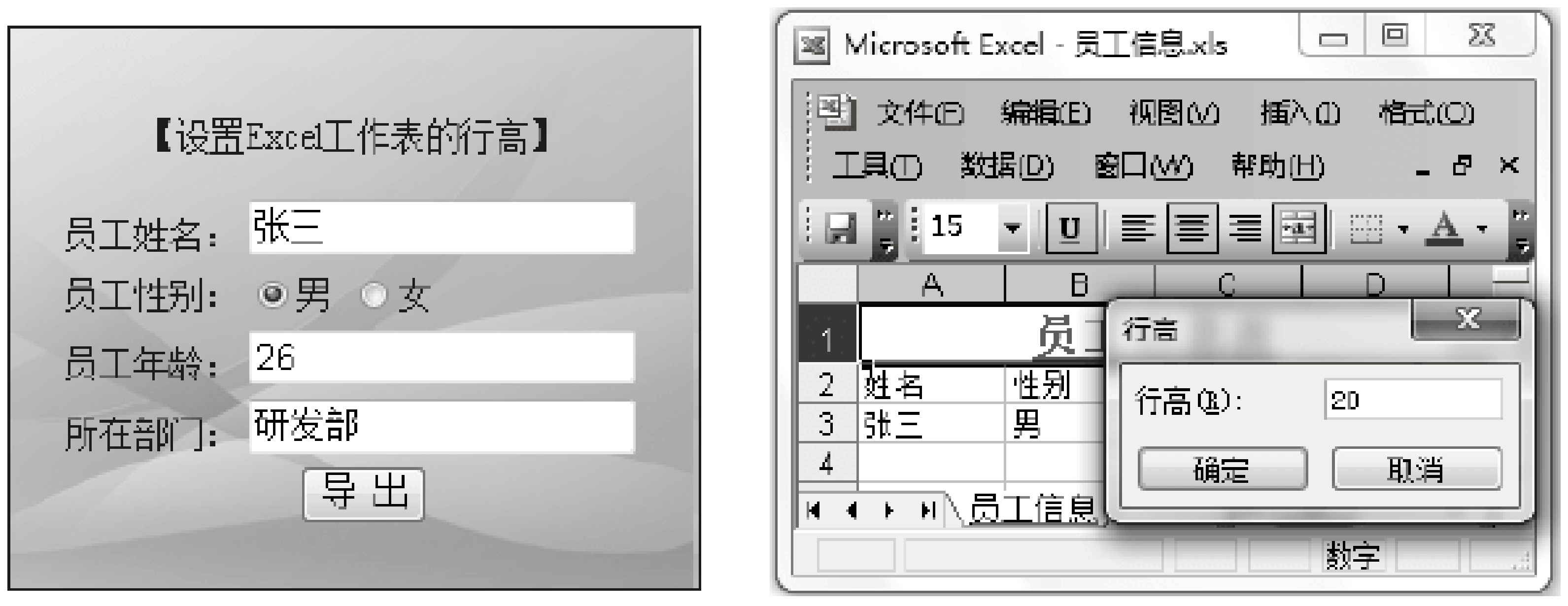


图 21.11　设置 Excel 工作表的行高

关键技术

设置 Excel 工作表的行高，主要应用的是 jxl.write.WritableSheet 接口的 setRowView()方法。通过 setRowView()方法可以设置指定行 i 的高度，其语法结构如下：

```
public void setRowView(int row, int height,boolean collapsed) throws RowsExceededException;
```

参数说明

❶ row：指定要设置行高的行索引，索引从 0 开始。索引为 0 即表示 Excel 工作表的第一行。

❷ height：指定行的高度。

❸ collapsed：指定是否隐藏该行。可选参数，取值为 true 或 false。默认值为 false，即不隐藏。

设计过程

创建用于操作 Excel 的工具类 ExcelOperationUtil，在该类中编写用于向 Excel 工作表中添加员工信息的方法 addInfoToExcel()，在该方法中设置 Excel 工作表的“员工信息”标题行的行高，其关键代码如下：

```
public boolean addInfoToExcel(String name,String sex,String age,String dept){
 try {
        WritableWorkbook book = Workbook.createWorkbook(new File("E:\\test\\员工信息.xls"));
        WritableSheet sheet = book.createSheet("员工信息", 0);
        jxl.write.WritableFont font = new jxl.write.WritableFont(WritableFont.ARIAL, 15, WritableFont.BOLD, false,UnderlineStyle.SINGLE,
jxl.format.Colour.GREEN);                    //设置字体样式
        jxl.write.WritableCellFormat fontFormat = new jxl.write.WritableCellFormat(font);
        fontFormat.setAlignment(Alignment.CENTRE);
        Label label_title = new Label(0,0,"员工信息表",fontFormat);
        sheet.mergeCells(0,0,3,0);                  //合并第 1 行的第 1～4 个单元格
        sheet.setRowView(0, 400,false);             //设置第 1 行的行高
        Label label_name = new Label(0,1,"姓名");
        Label label_sex = new Label(1,1,"性别");
        Label label_age = new Label(2,1,"年龄");
```

```
        Label label_dept = new Label(3,1,"所在部门");
        Label name_value = new Label(0,2,name);
        Label sex_value = new Label(1,2,sex);
        Label age_value = new Label(2,2,age);
        Label dept_value = new Label(3,2,dept);
        sheet.addCell(label_title);
        sheet.addCell(label_name);
        sheet.addCell(label_sex);
        sheet.addCell(label_age);
        sheet.addCell(label_dept);
        sheet.addCell(name_value);
        sheet.addCell(sex_value);
        sheet.addCell(age_value);
        sheet.addCell(dept_value);
        book.write();
        book.close();
        return true;
} catch (Exception e) {
        System.out.println("异常信息："+e.getMessage());
        e.printStackTrace();
        return false;
}
}
```

秘笈心法

应用 jxl.write.WritableSheet 接口的 setRowView()方法设置行的高度时，所设置的 Excel 工作表的行高并不是 setRowView()方法中设置的高度值 height，而是把 height 值除以 20 后的值再设置为 Excel 工作表的行高。从本实例的代码中可以看出，setRowView()方法中设置的高度为 400，但实际上在该方法的具体实现时是除以 20 的。

实例 543　设置 Excel 工作表的列宽

光盘位置：光盘\MR\21\543

初级

实用指数：★★★

实例说明

本实例将介绍如何应用 JXL 组件，设置 Excel 工作表的列宽。运行本实例，如图 21.12 所示，输入员工信息，当单击“导出”按钮后，员工信息将被添加到 Excel 工作表中，并且设置每一列的列宽为 20。

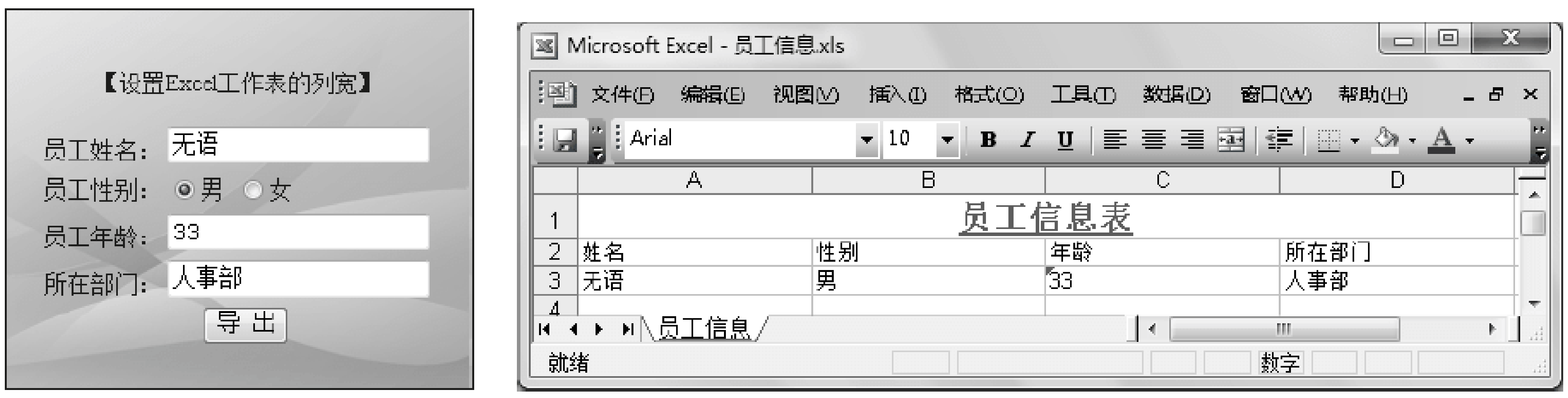


图 21.12　设置 Excel 工作表的列宽

关键技术

设置 Excel 工作表的列宽，主要应用的是 jxl.write.WritableSheet 接口的 setColumnView()方法。通过 setColumnView()方法可以设置指定第 i 列的宽度，其语法结构如下：

```
public void setColumnView(int col, int width)
```

参数说明

❶ col：指定要设置列宽的列索引，索引从 0 开始。索引为 0 时即表示 Excel 工作表的第一列。

❷ width：指定列的宽度。

设计过程

创建用于操作 Excel 的工具类 ExcelOperationUtil，在该类中编写用于向 Excel 工作表中添加员工信息的方法 addInfoToExcel()，在该方法中设置 Excel 工作表的列宽，其关键代码如下：

```
public boolean addInfoToExcel(String name,String sex,String age,String dept){
 try {
        WritableWorkbook book = Workbook.createWorkbook(new File("E:\\test\\员工信息.xls"));
        WritableSheet sheet = book.createSheet("员工信息", 0);
        //设置字体样式
        jxl.write.WritableFont font = new jxl.write.WritableFont(WritableFont.ARIAL, 15, WritableFont.BOLD, false,UnderlineStyle.SINGLE,
jxl.format.Colour.GREEN);
        jxl.write.WritableCellFormat fontFormat = new jxl.write.WritableCellFormat(font);
        fontFormat.setAlignment(Alignment.CENTRE);
        Label label_title = new Label(0,0,"员工信息表",fontFormat);
        sheet.mergeCells(0,0,3,0);//合并第一行的第 1～第 4 个单元格
        sheet.setRowView(0, 400,false);  //设置第 1 行的行高
        sheet.setColumnView(0, 20);      //设置第 1 列宽度
        sheet.setColumnView(1, 20);      //设置第 2 列宽度
        sheet.setColumnView(2, 20);      //设置第 3 列宽度
        sheet.setColumnView(3, 20);      //设置第 4 列宽度
        Label label_name = new Label(0,1,"姓名");
        Label label_sex = new Label(1,1,"性别");
        Label label_age = new Label(2,1,"年龄");
        Label label_dept = new Label(3,1,"所在部门");
        Label name_value = new Label(0,2,name);
        Label sex_value = new Label(1,2,sex);
        Label age_value = new Label(2,2,age);
        Label dept_value = new Label(3,2,dept);
        sheet.addCell(label_title);
        sheet.addCell(label_name);
        sheet.addCell(label_sex);
        sheet.addCell(label_age);
        sheet.addCell(label_dept);
        sheet.addCell(name_value);
        sheet.addCell(sex_value);
        sheet.addCell(age_value);
        sheet.addCell(dept_value);
        book.write();
        book.close();
        return true;
 } catch (Exception e) {
        System.out.println("异常信息："+e.getMessage());
        e.printStackTrace();
        return false;
 }
}
```

秘笈心法

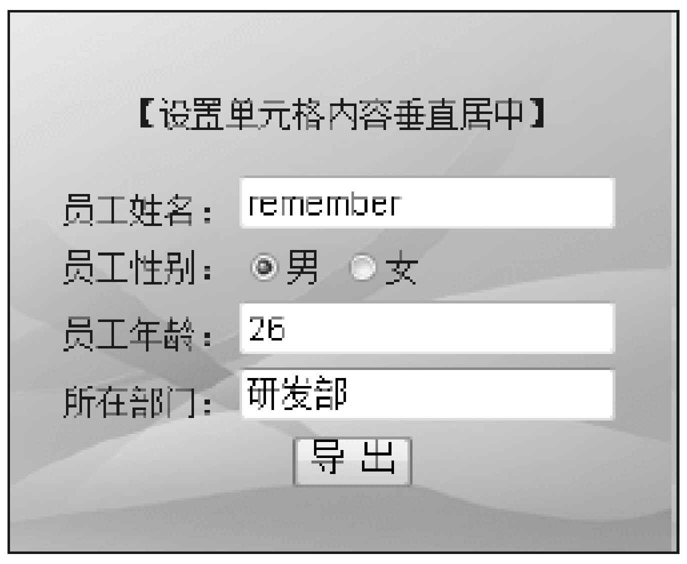
在实际应用中，由于 Excel 工作表的某一列的内容特别多，那么应用默认的列宽时，单元格的内容并不能完全显示。例如，某一列为日期时间类型的数据，如果宽度不够，将只显示“#”号。所以在应用 JXL 组件操作 Excel 时，应该及时应用 setColumnView()方法设置列宽。

实例 544	设置 Excel 工作表的单元格内容垂直居中 光盘位置：光盘\MR\21\544	初级 实用指数：★★★

实例说明

本实例将介绍如何应用 JXL 组件，设置 Excel 工作表的单元格内容垂直居中显示。运行本实例，如图 21.13 所示，输入员工信息，当单击“导出”按钮后，员工信息将被添加到 Excel 工作表中，并且设置“员工信息表”

标题垂直居中显示。



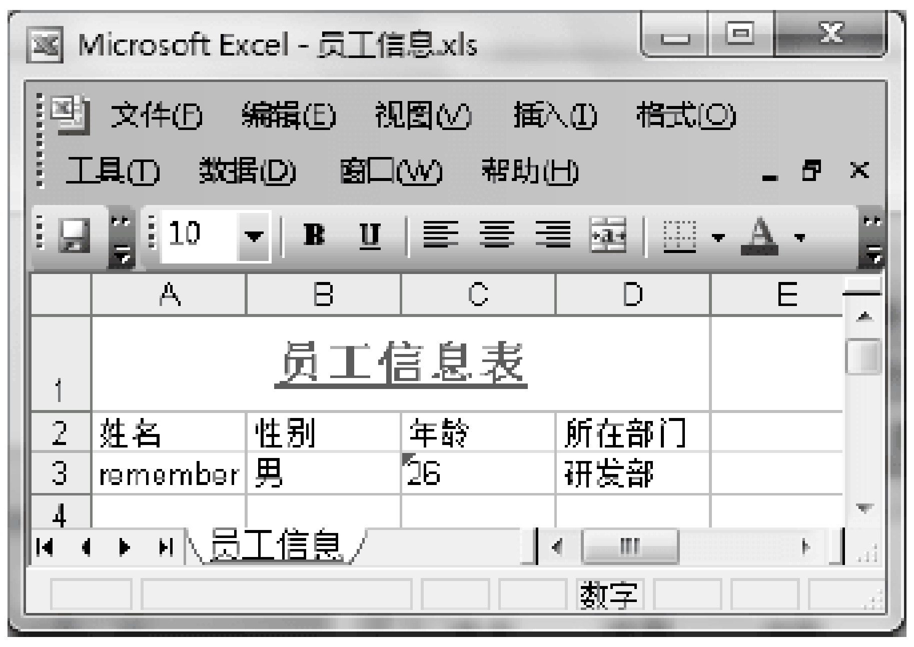


图 21.13 设置 Excel 工作表的单元格内容垂直居中显示

关键技术

设置 Excel 工作表单元格内容垂直居中显示，主要应用的是 jxl.write.WritableCellFormat 类的 setVertical Alignment()方法。WritableCellFormat 用于格式化单元格的样式，如单元格的字体样式、单元格边框样式、背景颜色、单元格凹陷、单元格锁定、单元格内容水平居中以及单元格内容垂直居中等。setVerticalAlignment()方法的语法结构如下：

```
public void setVerticalAlignment(VerticalAlignment va) throws WriteException
```

参数说明

va：参数类型为 jxl.format.VerticalAlignment。VerticalAlignment 专门用于设置单元格内容的垂直对齐方式，对齐方式的参数为静态参数，因此可以直接用 VerticalAlignment 类名进行调用，如 VerticalAlignment.TOP（居顶）、VerticalAlignment.CENTER（居中）、VerticalAlignment.BUTTOM（居底）等。

设计过程

创建用于操作 Excel 的工具类 ExcelOperationUtil，在该类中编写用于向 Excel 工作表中添加员工信息的方法 addInfoToExcel()，在该方法中设置“员工信息表”标题垂直居中显示，其关键代码如下：

```
public boolean addInfoToExcel(String name,String sex,String age,String dept){
 try {
        WritableWorkbook book = Workbook.createWorkbook(new File("E:\\test\\员工信息.xls"));
        WritableSheet sheet = book.createSheet("员工信息", 0);
        jxl.write.WritableFont font = new jxl.write.WritableFont(WritableFont.ARIAL, 15, WritableFont.BOLD, false,UnderlineStyle.SINGLE,
jxl.format.Colour.GREEN);                                       //设置字体样式
        jxl.write.WritableCellFormat fontFormat = new jxl.write.WritableCellFormat(font);
        fontFormat.setAlignment(Alignment.CENTRE);
        fontFormat.setVerticalAlignment(VerticalAlignment.JUSTIFY);     //设置单元格内容垂直居中显示
        Label label_title = new Label(0,0,"员工信息表",fontFormat);
        sheet.mergeCells(0,0,3,0);                                    //合并第 1 行的第 1～第 4 个单元格
        sheet.setRowView(0, 600,false);                               //设置第 1 行的行高
        Label label_name = new Label(0,1,"姓名");
        Label label_sex = new Label(1,1,"性别");
        Label label_age = new Label(2,1,"年龄");
        Label label_dept = new Label(3,1,"所在部门");
        Label name_value = new Label(0,2,name);
        Label sex_value = new Label(1,2,sex);
        Label age_value = new Label(2,2,age);
        Label dept_value = new Label(3,2,dept);
        sheet.addCell(label_title);
        sheet.addCell(label_name);
        sheet.addCell(label_sex);
        sheet.addCell(label_age);
        sheet.addCell(label_dept);
        sheet.addCell(name_value);
        sheet.addCell(sex_value);
        sheet.addCell(age_value);
        sheet.addCell(dept_value);
        book.write();
        book.close();
        return true;
```

```
} catch (Exception e) {
        System.out.println("异常信息："+e.getMessage());
        e.printStackTrace();
        return false;
}
}
```

秘笈心法

在实际应用中，根据需求可能需要设置单元格的内容垂直显示样式，这就需要应用 jxl.write.WritableCellFormat 类的 setVerticalAlignment()方法进行设置。

实例 545 设置 Excel 工作表的单元格内容自动换行

光盘位置：光盘\MR\21\545

初级

实用指数：★★★

实例说明

本实例将介绍如何应用 JXL 组件，设置 Excel 工作表的单元格内容自动换行。运行本实例，如图 21.14 所示，输入员工信息，当单击“导出”按钮后，员工信息将被添加到 Excel 工作表中，并且设置“所在部门”单元格内容自动换行。

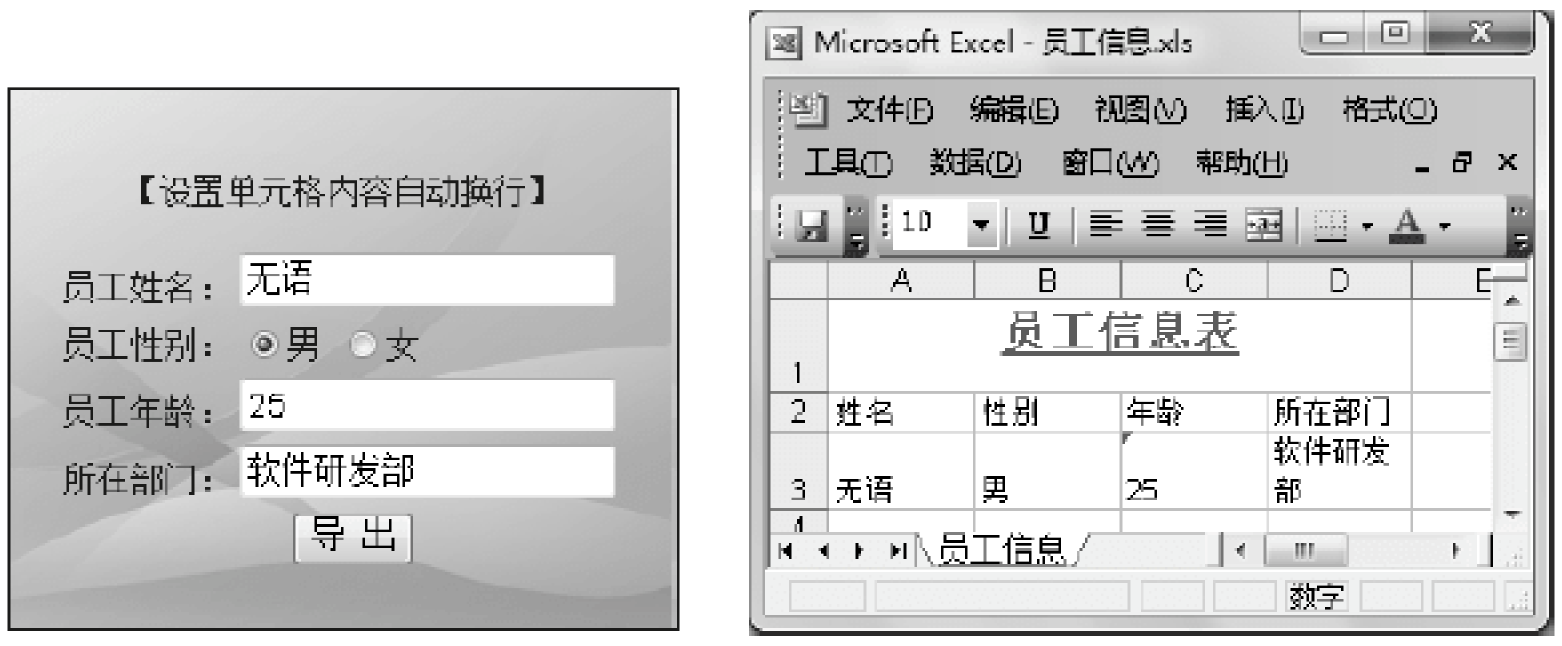


图 21.14 设置 Excel 工作表的单元格内容自动换行

关键技术

设置 Excel 工作表单元格内容自动换行，主要应用的是 jxl.write.WritableCellFormat 类的 setWrap()方法。setWrap()方法的语法结构如下：

```
public void setWrap(boolean w) throws WriteException
```

参数说明

w：指定是否换行。取值为 true 或 false。

设计过程

创建用于操作 Excel 的工具类 ExcelOperationUtil，在该类中编写用于向 Excel 工作表中添加员工信息的方法 addInfoToExcel()，在该方法中设置了“所在部门”单元格的自动换行，其关键代码如下：

```
public boolean addInfoToExcel(String name,String sex,String age,String dept){
 try {
        WritableWorkbook book = Workbook.createWorkbook(new File("E:\\test\\员工信息.xls"));
        WritableSheet sheet = book.createSheet("员工信息", 0);
        jxl.write.WritableFont font = new jxl.write.WritableFont(WritableFont.ARIAL, 15, WritableFont.BOLD, false,UnderlineStyle.SINGLE,
jxl.format.Colour.GREEN);                                       //设置字体样式
        jxl.write.WritableCellFormat fontFormat = new jxl.write.WritableCellFormat(font);
        fontFormat.setAlignment(Alignment.CENTRE);
        fontFormat.setVerticalAlignment(VerticalAlignment.JUSTIFY);     //设置单元格内容垂直居中显示
        contentFormat.setWrap(true);
        Label label_title = new Label(0,0,"员工信息表",fontFormat);
        sheet.mergeCells(0,0,3,0);                              //合并第 1 行的第 1～第 4 个单元格
        sheet.setRowView(0, 600,false);                         //设置第 1 行的行高
        Label label_name = new Label(0,1,"姓名");
```

```
        Label label_sex = new Label(1,1,"性别");
        Label label_age = new Label(2,1,"年龄");
        Label label_dept = new Label(3,1,"所在部门");
        Label name_value = new Label(0,2,name);
        Label sex_value = new Label(1,2,sex);
        Label age_value = new Label(2,2,age);
        Label dept_value = new Label(3,2,dept);
        sheet.addCell(label_title);
        sheet.addCell(label_name);
        sheet.addCell(label_sex);
        sheet.addCell(label_age);
        sheet.addCell(label_dept);
        sheet.addCell(name_value);
        sheet.addCell(sex_value);
        sheet.addCell(age_value);
        sheet.addCell(dept_value);
        book.write();
        book.close();
        return true;
} catch (Exception e) {
        System.out.println("异常信息："+e.getMessage());
        e.printStackTrace();
        return false;
}
}
```

秘笈心法

在 Excel 工作表中，默认添加的单元格内容是不换行的，当某一个单元格的内容超过默认的单元格宽度时，这些内容将显示在下一个单元格中。所以，如果不希望单元格内容显示在下一个单元格中，此时可以考虑应用 setWrap()方法设置单元格内容自动换行。

实例 546　设置 Excel 工作表的单元格样式

光盘位置：光盘\MR\21\546

初级　实用指数：★★★

实例说明

本实例将介绍如何应用 JXL 组件，设置 Excel 工作表的单元格样式。运行本实例，如图 21.15 所示，输入员工信息，当单击“导出”按钮后，员工信息将被添加到 Excel 工作表中，并且设置“员工信息表”标题的单元格样式。

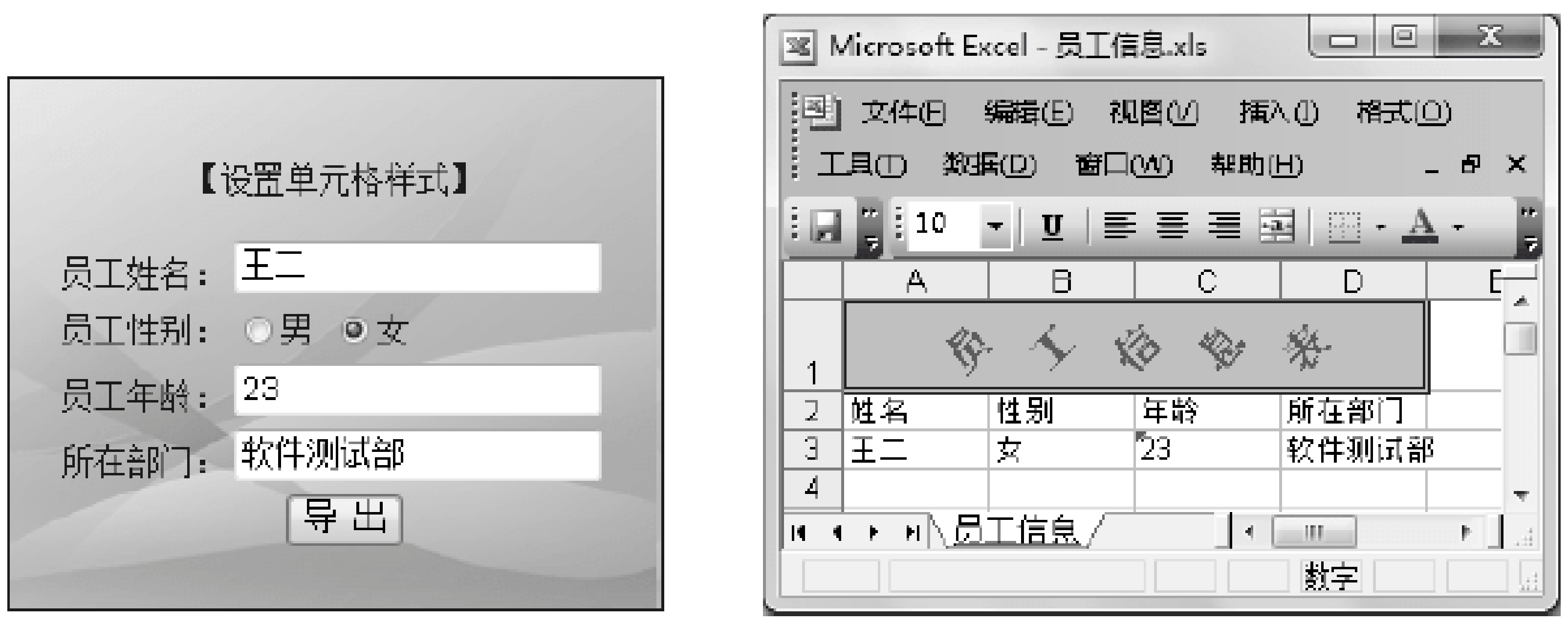


图 21.15　设置 Excel 工作表的单元格样式

关键技术

本实例在设置 Excel 工作表单元格样式时，主要应用的是 jxl.write.WritableCellFormat 类的 setBackground()方法、setBorder()方法和 setOrientation()方法。下面对这几个方法进行详细介绍。

（1）setBackground()方法

该方法主要用于设置单元格的背景颜色，其语法结构如下：

```
public void setBackground(Colour colour, Pattern pattern) throws WriteException
```

参数说明

❶ colour：设置单元格的背景颜色。颜色的参数值为 static 类型，所以可以直接应用 jxl.format.Colour 类名调用。

❷ pattern：此参数为可选参数，用于设置单元格背景图案。图案的参数值为 static 类型，所以可以直接应用 jxl.format.Pattern 类名调用。

（2）setBorder()方法

该方法用于设置单元格的边框样式，其语法结构如下：

```
public void setBorder(Border border, BorderLineStyle ls, Colour colour) throws WriteException
```

参数说明

❶ border：指定设置的是单元格四周的哪一个方向的边框。主要取值包括 Border.ALL（四周）、Border.LEFT（左边框）、Border.RIGHT（右边框）、Border.TOP（上边框）和 Border.BUTTOM（下边框）。

❷ ls：设置单元格边框的线条样式。此参数值可以应用 jxl.format.BorderLineStyle 类名直接调用，如 BorderLineStyle.THICK（加粗）、BorderLineStyle.DASHED（虚线）、BorderLineStyle.THIN（细线）、BorderLineStyle.DOUBLE（双线）等。

❸ colour：可选参数，用于设置单元格边框颜色。

（3）setOrientation()方法

该方法用于设置单元格文字的方向，其语法结构如下：

```
public void setOrientation(Orientation orientation) throws WriteException
```

参数说明

orientation：参数类型为 jxl.format.Orientation，用于设置文字的方向，如 Orientation.PUSH_45（斜向上 45°）、Orientation.PUSH_90（向上 90°）、Orientation.MINUS_45（斜向下 45°）、Orientation.MINUS_90（向下 90°）等。

设计过程

创建用于操作 Excel 的工具类 ExcelOperationUtil，在该类中编写用于向 Excel 工作表中添加员工信息的方法 addInfoToExcel()，在该方法中设置了“员工信息表”标题单元格的样式，其关键代码如下：

```
public boolean addInfoToExcel(String name,String sex,String age,String dept){
 try {
        WritableWorkbook book = Workbook.createWorkbook(new File("E:\\test\\员工信息.xls"));
        WritableSheet sheet = book.createSheet("员工信息", 0);
        //设置字体样式
        jxl.write.WritableFont font = new jxl.write.WritableFont(WritableFont.ARIAL, 15, WritableFont.BOLD, false,UnderlineStyle.NO_UNDERLINE,
jxl.format.Colour.GREEN);
        jxl.write.WritableCellFormat cellFormat = new jxl.write.WritableCellFormat(font);
        cellFormat.setAlignment(Alignment.CENTRE);
        cellFormat.setVerticalAlignment(VerticalAlignment.JUSTIFY);                    //设置单元格内容两端对齐
        cellFormat.setBackground(Colour.GRAY_25);                                      //背景颜色
        cellFormat.setBorder(Border.ALL, BorderLineStyle.DOUBLE,Colour.DARK_GREEN);    //边框样式
        cellFormat.setOrientation(Orientation.PLUS_45);                                //文字方向
        Label label_title = new Label(0,0,"员工信息表",cellFormat);
        sheet.mergeCells(0,0,3,0);                                                     //合并第 1 行的第 1～4 个单元格

        sheet.setRowView(0, 600,false);                                                //设置第 1 行的行高
        Label label_name = new Label(0,1,"姓名");
        Label label_sex = new Label(1,1,"性别");
        Label label_age = new Label(2,1,"年龄");
        Label label_dept = new Label(3,1,"所在部门");
        Label name_value = new Label(0,2,name);
        Label sex_value = new Label(1,2,sex);
        Label age_value = new Label(2,2,age);
        Label dept_value = new Label(3,2,dept);
        sheet.addCell(label_title);
        sheet.addCell(label_name);
        sheet.addCell(label_sex);
        sheet.addCell(label_age);
        sheet.addCell(label_dept);
```

```
            sheet.addCell(name_value);
            sheet.addCell(sex_value);
            sheet.addCell(age_value);
            sheet.addCell(dept_value);
            book.write();
            book.close();
            return true;
    } catch (Exception e) {
            System.out.println("异常信息："+e.getMessage());
            e.printStackTrace();
            return false;
    }
}
```

秘笈心法

在实际开发应用中，为了使打印出的 Excel 工作表具有特殊效果，经常需要设置 Excel 工作表的单元格样式。所以，应该掌握应用 jxl.write.WritableCellFormat 类设置单元格样式的方法。

实例 547　向 Excel 工作表中插入图片

光盘位置：光盘\MR\21\547

中级　实用指数：★★★

实例说明

本实例将介绍如何应用 JXL 组件，向 Excel 工作表中插入图片。运行本实例，如图 21.16 所示，输入员工信息，当单击“导出”按钮后，员工信息将被添加到 Excel 工作表中，并且向 Excel 工作表中插入图片。

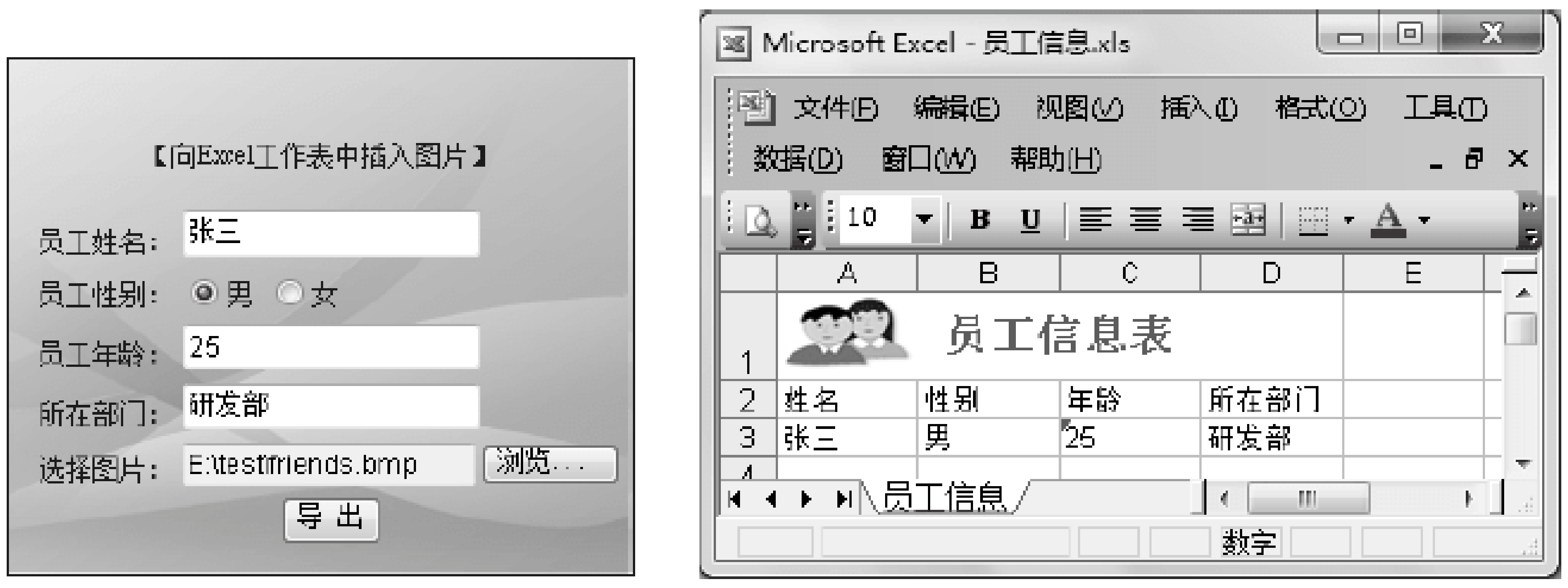


图 21.16　向 Excel 工作表中插入图片

关键技术

应用 JXL 组件可以向 Excel 工作表中插入图片。首先在插入图片之前应该创建一个图片对象，该图片对象是由 jxl.write.WritableImage 类创建的，然后调用 WritableSheet 工作表对象的 addImage(WritableImage image)方法执行插入即可。WritableImage 类的构造方法结构如下：

```
public WritableImage(double x, double y, double width, double height,byte imageData)
```

参数说明

❶ x：指定图片所在位置的列索引。

❷ y：指定图片所在位置的行索引。

❸ width：指定图片所占的横向单元格个数，也就是图片跨越了几列。

❹ height：指定图片所占的纵向单元格个数，也就是图片跨越了几行。

❺ imageData：表示图片数据的 byte 数组。

设计过程

创建用于操作 Excel 的工具类 ExcelOperationUtil，在该类中编写用于向 Excel 工作表中添加员工信息的方法 addInfoToExcel()，并且向 Excel 工作表中插入了图片，其关键代码如下：

```
public boolean addInfoToExcel(String name,String sex,String age,String dept, String imgPath){
try {
        WritableWorkbook book = Workbook.createWorkbook(new File("E:\\test\\员工信息.xls"));
        WritableSheet sheet = book.createSheet("员工信息", 0);
        jxl.write.WritableFont font = new jxl.write.WritableFont(WritableFont.ARIAL, 15, WritableFont.BOLD, false,UnderlineStyle.NO_UNDERLINE,
jxl.format.Colour.GREEN);                                    //设置字体样式
        jxl.write.WritableCellFormat cellFormat = new jxl.write.WritableCellFormat(font);
        cellFormat.setAlignment(Alignment.CENTRE);
        cellFormat.setVerticalAlignment(VerticalAlignment.CENTRE);
        File file = new File(imgPath);                       //根据图片路径创建 File 对象
        FileInputStream fs = new FileInputStream(file);      //创建字节输入流，用于读取图片字节数据
        byte[] bytes = new byte[fs.available()];             //根据图片的字节数，创建字节数组
        fs.read(bytes);                                      //读取图片字节数据保存到字节数组中
        WritableImage image = new WritableImage(0,1,1,2,bytes);  //创建 WritableImage 对象
        Label label_title = new Label(0,0,"员工信息表",cellFormat);
        sheet.mergeCells(0,0,4,0);                           //合并第 1 行的第 1～4 个单元格
        sheet.setRowView(0, 600,false);                      //设置第 1 行的行高
        Label label_name = new Label(1,1,"姓名");
        Label label_sex = new Label(2,1,"性别");
        Label label_age = new Label(3,1,"年龄");
        Label label_dept = new Label(4,1,"所在部门");
        Label name_value = new Label(1,2,name);
        Label sex_value = new Label(2,2,sex);
        Label age_value = new Label(3,2,age);
        Label dept_value = new Label(4,2,dept);
        sheet.addCell(label_title);
        sheet.addCell(label_name);
        sheet.addCell(label_sex);
        sheet.addCell(label_age);
        sheet.addCell(label_dept);
        sheet.addCell(name_value);
        sheet.addCell(sex_value);
        sheet.addCell(age_value);
        sheet.addCell(dept_value);
        sheet.addImage(image);
        book.write();
        book.close();
        return true;
} catch (Exception e) {
        System.out.println("异常信息："+e.getMessage());
        e.printStackTrace();
        return false;
}
}
```

秘笈心法

jxl.write.WritableImage 类还包含另一个构造方法，其语法结构如下：

```
public WritableImage(double x, double y, double width, double height,File image)
```

从构造方法中可以看出，前 4 个参数与本实例讲解的是相同的。只有最后一个参数为 java.io.File 类型，用来表示图片路径的 File 对象。但是应用这个构造方法创建 WritableImage 对象时，支持的图片格式只有 png 格式。所以，如果希望插入其他格式的图片，不要使用该构造方法，否则会抛出异常。

实例 548	将数据库数据导出到 Excel 光盘位置：光盘\MR\21\548	高级 实用指数：★★★

实例说明

本实例将介绍如何应用 JXL 组件，读取数据库表的数据并导出到 Excel 文件中。运行本实例，如图 21.17 所示，单击“导出”按钮后，所有员工信息将被导出到 Excel 文件中。

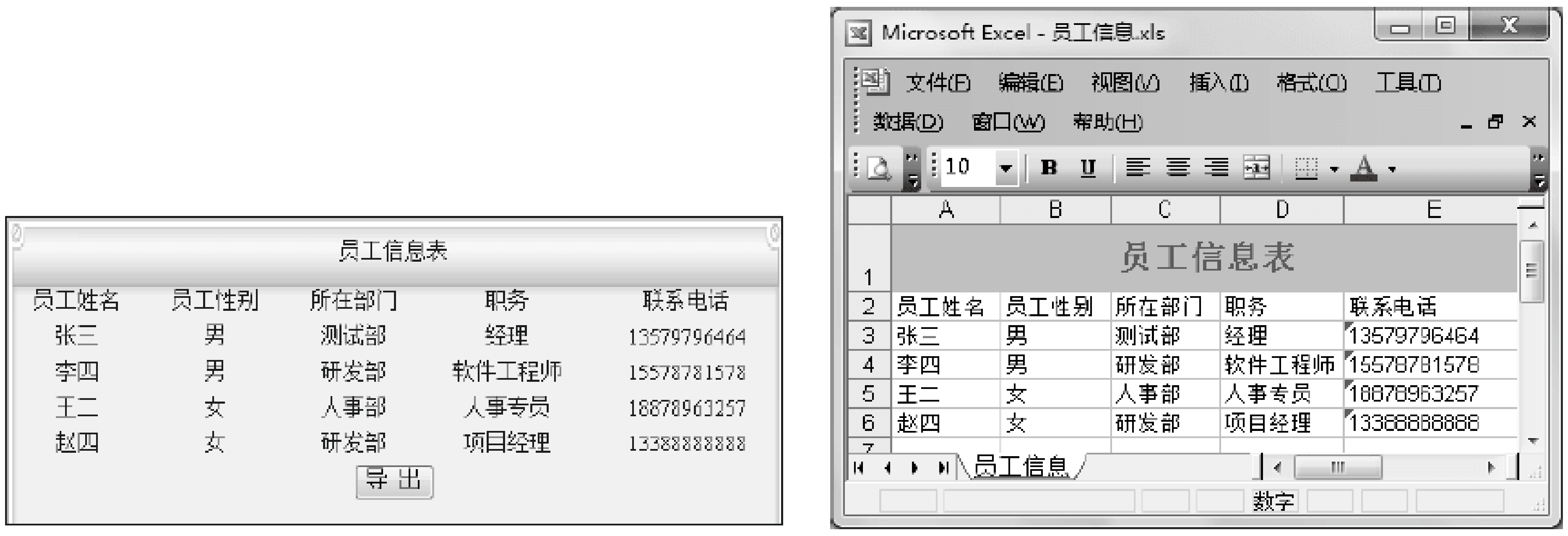


图 21.17　将数据库员工信息表的数据导出到 Excel 中

关键技术

实现了前面一系列对 Excel 操作的实例之后，再来实现本实例其实很简单，只是将读取出的数据库的数据再导出到 Excel 文件中。首先将读取出的数据库表数据封装到一个 List 集合中，然后循环遍历该 List 集合，再应用 JXL 组件将这些数据写入 Excel 文件。具体向 Excel 中如何写数据，在前面的一些例子中已经讲解了，此处不再赘述。

设计过程

（1）创建用于封装员工信息表数据的 JavaBean 类 Employee，其关键代码如下：

```
public class Employee {
        private int id;
        private String name;
        private String sex;
        private String dept;
        private String duty;
        private String telephone;
        ……//此处省略了属性的 getXXX()方法和 setXXX()方法
}
```

（2）创建用于操作 Excel 的工具类 ExcelOperationUtil，在该类中编写用于向 Excel 工作表中添加员工信息的方法 readDataToExcelFile()，参数 list 封装了从数据库表中读取数据的 JavaBean 对象 Employee，其关键代码如下：

```
public boolean readDataToExcelFile(List<Employee> list){
 try {
        WritableWorkbook book = Workbook.createWorkbook(new File("E:\\test\\员工信息.xls"));
        WritableSheet sheet = book.createSheet("员工信息", 0);
        jxl.write.WritableFont font = new jxl.write.WritableFont(WritableFont.ARIAL, 15, WritableFont.BOLD, false,UnderlineStyle.NO_UNDERLINE,
jxl.format.Colour.GREEN);                                       //设置字体样式
        jxl.write.WritableCellFormat cellFormat = new jxl.write.WritableCellFormat(font);
        cellFormat.setAlignment(Alignment.CENTRE);
        cellFormat.setVerticalAlignment(VerticalAlignment.CENTRE);      //设置单元格内容两端对齐
        cellFormat.setBackground(Colour.GRAY_25);                       //设置背景颜色
        Label label_title = new Label(0,0,"员工信息表",cellFormat);
        sheet.mergeCells(0,0,4,0);                                      //合并第 1 行的第 1～5 个单元格
        sheet.setRowView(0, 600,false);                                 //设置第 1 行的行高
        Label label_name = new Label(0,1,"员工姓名");
        Label label_sex = new Label(1,1,"员工性别");
        Label label_dept = new Label(2,1,"所在部门");
        Label label_duty = new Label(3,1,"职务");
        Label label_telephone = new Label(4,1,"联系电话");
        sheet.setColumnView(4, 15);                                     //设置列宽
        sheet.addCell(label_title);
        sheet.addCell(label_name);
        sheet.addCell(label_sex);
        sheet.addCell(label_dept);
        sheet.addCell(label_duty);
        sheet.addCell(label_telephone);
        for(int i=0;i<list.size();i++){                                 //遍历保存员工信息对象的集合，将所有员工信息导出到 Excel
```

```
                Employee emp = (Employee)list.get(i);
                Label name_value = new Label(0,i+2,emp.getName());
                Label sex_value = new Label(1,i+2,emp.getSex());
                Label dept_value = new Label(2,i+2,emp.getDept());
                Label duty_value = new Label(3,i+2,emp.getDuty());
                Label telephone_value = new Label(4,i+2,emp.getTelephone());
                sheet.addCell(name_value);
                sheet.addCell(sex_value);
                sheet.addCell(dept_value);
                sheet.addCell(duty_value);
                sheet.addCell(telephone_value);
            }
            book.write();
            book.close();
            return true;
        } catch (Exception e) {
            System.out.println("异常信息："+e.getMessage());
            e.printStackTrace();
            return false;
        }
    }
```

说明： 由于读取数据库的代码比较简单，所以此处省略了这一部分内容。具体代码可以查看本书附赠的光盘。

秘笈心法

在实际应用中，经常会将数据库中某个表的数据导出到 Excel 文件中，然后再应用 Excel 将这些数据打印出来，以便于查看，如员工信息、订单信息或一些办公文件的表单信息。所以，实现将数据库的数据导出到 Excel 文件中是非常有必要的。

实例 549 读取 Excel 中的数据和图片并保存到数据库 高级

光盘位置：光盘\MR\21\549

实用指数：★★★

实例说明

本实例将介绍如何应用 JXL 组件，读取 Excel 工作表中的数据以及图片，然后将这些数据保存到数据库中。运行本实例，如图 21.18 所示，单击“保存到数据库”按钮后，“员工信息.xls”工作表的数据将被保存到数据库中，并且保存了该 Excel 工作表中的图片。



图 21.18 读取 Excel 工作表中的数据和图片并保存到数据库

关键技术

读取 Excel 文件中的数据，主要应用到了 jxl.Workbook 类和 jxl.Sheet 接口。下面对 Workbook 类和 Sheet 接口进行详细介绍。

（1）Workbook 类

在读取 Excel 文件时，首先需要获取 jxl.Workbook 工作簿对象，然后再从 Workbook 对象中获取工作表对象。获取 Workbook 对象的语法结构如下：

```
Workbook book = Workbook.getWorkbook(InputStream is)
```

参数说明

is：表示字节输入流对象，用于从 Excel 文件中读取字节流数据。

（2）Sheet 接口

Sheet 是一个接口，该接口中提供了一系列用于读取 Excel 工作表的 getXXX()方法和其他方法。该接口中的所有方法是在 jxl.read.biff.SheetImpl 类中实现的。工作表 Sheet 对象是由 Workbook 工作簿获取的，语法结构如下：

```
Sheet sheet = book.getSheet(int index)
```

参数说明

❶ book：表示工作簿 Workbook 对象。

❷ index：表示 Excel 工作表的索引。索引值从 0 开始。

getSheet()方法还有另一个重载方法，其语法结构如下：

```
Sheet sheet = book.getSheet(String name)
```

参数说明

❶ book：表示工作簿 Workbook 对象。

❷ name：表示 Excel 工作表的名称。

Sheet 接口中定义了一些常用的用于获取 Excel 工作表相关信息的方法，这些方法以及说明如表 21.1 所示。

表 21.1　Sheet 接口的方法

方　　法	说　　明
public int getRows()	用于获取 Excel 工作表中数据的行数
public int getColumns()	用于获取 Excel 工作表中数据的列数
public Cell getCell(int column,int row)	用于获取 Excel 工作表中指定的单元格对象。参数 column 用于指定单元格所在的列索引，参数 row 用于指定单元格所在的行索引
public Cell[] getColumn(int col)	返回 Excel 工作表某一列的所有单元格对象数组。参数 col 为指定的列索引
public Image getDrawing(int i);	用于获取 Excel 工作表中的图片对象。参数 i 为 Excel 工作表中插入图片的索引。方法返回 jxl.Image 类型的对象
public int getNumberOfImages();	返回 Excel 工作表中图片的个数。当一个工作表中包含多张图片时，需要在循环中应用此方法，结合使用 getDrawing()方法来获取所有图片

设计过程

（1）创建用于封装员工信息表数据的 JavaBean 类 Employee。注意，在该类中添加了一个用于保存图片的 jxl.Image 属性，关键代码如下：

```
import jxl.Image;
public class Employee {
	private int id;
	private String name;
	private String sex;
	private String dept;
	private String duty;
	private String telephone;
	private Image image; //属性类型为 jxl.Image
	……//此处省略了属性的 getXXX()方法和 setXXX()方法
}
```

（2）创建用于操作 Excel 文件的工具类 ExcelOperationUtil，在该类中编写用于读取 Excel 工作表数据的方法 readExcelData()，参数 filePath 是指用户选择的 Excel 文件的路径。在 readExcelData()方法中，将读取出的数据保存在 JavaBean 对象 Employee 中，然后将所有的 Employee 对象再添加到 List 集合中，最后使用 readExcelData()方法返回这个 List 集合，关键代码如下：

```
public List<Employee> readExcelData(String filePath){
 List<Employee> list = new ArrayList<Employee>();
 try {
```

```
        File xlsFile = new File(filePath);
        FileInputStream fs = new FileInputStream(xlsFile);
        Workbook book = Workbook.getWorkbook(fs);                   //获取工作簿对象
        Sheet sheet = book.getSheet(0);                             //获取工作表对象
        int rows = sheet.getRows();                                 //获取工作表中的数据行数
        for(int i=2;i<rows;i++){                                    //循环 Excel 工作表的行，并读取单元格数据
            Image img = sheet.getDrawing(0);                        //获取工作表中的图片对象
            Employee emp = new Employee();
            String name = sheet.getCell(0, i).getContents();
            String sex = sheet.getCell(1, i).getContents();
            String dept = sheet.getCell(2, i).getContents();
            String duty = sheet.getCell(3, i).getContents();
            String telephone = sheet.getCell(4, i).getContents();
            emp.setName(name);
            emp.setSex(sex);
            emp.setDept(dept);
            emp.setDuty(duty);
            emp.setTelephone(telephone);
            emp.setImage(img);                                      //将图片对象添加到 Employee 中
            list.add(emp);
        }
        return list;
} catch (Exception e) {
        System.out.println("异常信息："+e.getMessage());
        e.printStackTrace();
        return null;
}
}
```

（3）创建用于操作数据库的 DAO 类 EmpDao，编写用于保存数据的方法 saveEmpFromExcel()，参数 list 为封装从 Excel 文件中读取出的数据，关键代码如下：

```
public boolean saveEmpFromExcel(List<Employee> list) throws SQLException{
        Connection conn = null;
        try{
            conn = DBCon.getConn();                                 //获取数据库连接
            String sql = "insert into tb_emp(name,sex,dept,duty,telephone,img) values(?,?,?,?,?,?);";  //插入数据的 insert 语句
            PreparedStatement stmt = conn.prepareStatement(sql);    //创建 PreparedStatement 对象
            for(Employee emp:list){                                 //遍历封装数据的集合
                stmt.setString(1, emp.getName());
                stmt.setString(2, emp.getSex());
                stmt.setString(3, emp.getDept());
                stmt.setString(4, emp.getDuty());
                stmt.setString(5, emp.getTelephone());
                stmt.setBytes(6, emp.getImage().getImageData());    //读取图片的原始字节数据
                stmt.addBatch();                 //将一组参数添加到此 PreparedStatement 对象的批处理命令中
            }
            stmt.executeBatch();                 //将以上一批命令提交给数据库执行
            return true;
        }catch(Exception ex){
            System.out.println("保存员工数据异常："+ex.getMessage());
            ex.printStackTrace();
            return false;
        }finally{
            conn.close();
        }
}
```

秘笈心法

在实际应用中，有些情况下需要将一个 Excel 工作表的数据保存到数据库中，当这个 Excel 工作表中包含大量的数据时，可以考虑在程序中应用 JXL 组件来读取 Excel 数据。

实例 550 **设置 Excel 工作表简单的打印属性** 初级

光盘位置：光盘\MR\21\550 实用指数：★★★

实例说明

本实例将介绍如何应用 JXL 组件设置 Excel 工作表的打印属性，主要设置页眉和页脚。运行本实例，如图 21.19 所示，单击“导出”按钮后，所有员工信息将被导出到 Excel 文件中，并设置了打印属性，包括设置页眉、页脚、打印方向、页面纸张大小等。如图 21.20 所示为在打印预览中查看的页脚属性。

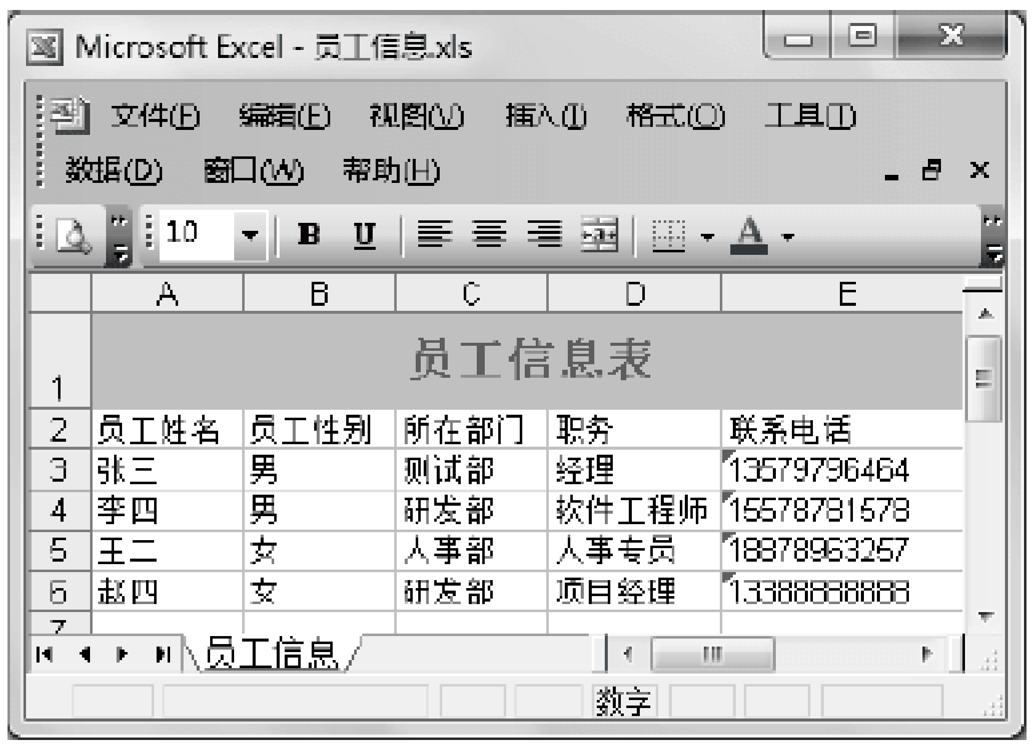


图 21.19 导出的员工信息数据

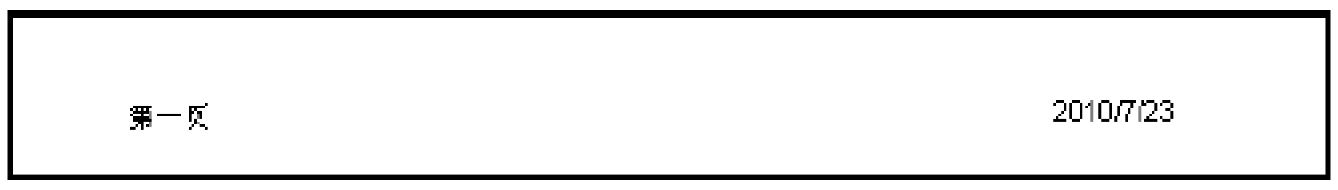


图 21.20 设置页脚

关键技术

应用 JXL 组件可以设置 Excel 工作表的打印属性，主要是应用 jxl.write.WritableSheet 中的 setHeader()方法、setFooter()方法和 setPageSetup()方法。下面对这几个方法进行详细介绍。

（1）setHeader()方法

该方法用于设置 Excel 工作表打印时的页眉，其语法结构如下：

```
public void setHeader(String left, String center, String right)
```

参数说明

❶ left：设置页眉左侧的内容。

❷ center：设置页眉中间的内容。

❸ right：设置页眉右侧的内容。

（2）setFooter()方法

该方法用于设置 Excel 工作表打印时的页脚，其语法结构如下：

```
public void setFooter(String left, String center, String right)
```

参数说明

❶ left：设置页脚左侧的内容。

❷ center：设置页脚中间的内容。

❸ right：设置页脚右侧的内容。

（3）setPageSetup()方法

该方法主要用于设置打印方向、打印纸张的大小和页眉页脚距打印纸的上下两端的距离，其语法结构如下：

```
public void setPageSetup(PageOrientation orientation, PaperSize size, double headerMargin, double footerMargin)
```

参数说明

❶ orientation：设置 Excel 工作表的打印方向。取值为 PageOrientation.LANDSCAPE（横向）和 PageOrientation.PORTRAIT（纵向）。

❷ size：可选参数，设置打印纸的大小，如 PaperSize.A2、PaperSize.A3、PaperSize.A4、PaperSize.A5 等。如果不设置此参数，则采用默认的 PaperSize.A4 值。

❸ headerMargin：可选参数，设置页眉距页面顶端的距离，单位为厘米。如果不设置此参数，默认值为 0.5。

❹ footerMargin：可选参数，设置页脚距页面底端的距离，单位为厘米。如果不设置此参数，默认值为 0.5。

设计过程

创建用于操作 Excel 的工具类 ExcelOperationUtil，在该类中编写用于向 Excel 工作表中添加员工信息的方法 readDataToExcelFile()，参数 list 封装了从数据库表中读取数据的 JavaBean 对象 Employee。在该方法中，导入员工数据的同时设置了打印属性，关键代码如下：

```
public boolean readDataToExcelFile(List<Employee> list){
  try {
        WritableWorkbook book = Workbook.createWorkbook(new File("E:\\test\\员工信息.xls"));
        WritableSheet sheet = book.createSheet("员工信息", 0);
        //设置字体样式
        jxl.write.WritableFont font = new jxl.write.WritableFont(WritableFont.ARIAL, 15, WritableFont.BOLD, false,UnderlineStyle.NO_UNDERLINE,
jxl.format.Colour.GREEN);
        jxl.write.WritableCellFormat cellFormat = new jxl.write.WritableCellFormat(font);
        cellFormat.setAlignment(Alignment.CENTRE);
        cellFormat.setVerticalAlignment(VerticalAlignment.CENTRE);          //设置单元格内容两端对齐
        cellFormat.setBackground(Colour.GRAY_25);                           //设置背景颜色
        Label label_title = new Label(0,0,"员工信息表",cellFormat);
        sheet.mergeCells(0,0,4,0);                                          //合并第 1 行的第 1～5 个单元格
        sheet.setRowView(0, 600,false);                                     //设置第 1 行的行高
        Label label_name = new Label(0,1,"员工姓名");
        Label label_sex = new Label(1,1,"员工性别");
        Label label_dept = new Label(2,1,"所在部门");
        Label label_duty = new Label(3,1,"职务");
        Label label_telephone = new Label(4,1,"联系电话");
        sheet.setColumnView(4, 15);                                         //设置列宽
        sheet.addCell(label_title);
        sheet.addCell(label_name);
        sheet.addCell(label_sex);
        sheet.addCell(label_dept);
        sheet.addCell(label_duty);
        sheet.addCell(label_telephone);
        for(int i=0;i<list.size();i++){                                     //遍历保存员工信息对象的集合，将所有员工信息导出到 Excel
              Employee emp = (Employee)list.get(i);
              Label name_value = new Label(0,i+2,emp.getName());
              Label sex_value = new Label(1,i+2,emp.getSex());
              Label dept_value = new Label(2,i+2,emp.getDept());
              Label duty_value = new Label(3,i+2,emp.getDuty());
              Label telephone_value = new Label(4,i+2,emp.getTelephone());
              sheet.addCell(name_value);
              sheet.addCell(sex_value);
              sheet.addCell(dept_value);
              sheet.addCell(duty_value);
              sheet.addCell(telephone_value);
        }
        Calendar c = Calendar.getInstance();
        String date=c.get(c.YEAR)+"/"+(c.get(c.MONTH)+1)+"/"+c.get(c.DAY_OF_MONTH);
        sheet.setHeader("", "", "普通文件");                                //设置页眉
        sheet.setFooter("", "第一页", date);                                //设置页脚
        sheet.setPageSetup(PageOrientation.PORTRAIT, PaperSize.A4, 0.2, 0.2); ;  //设置打印纸的属性
        book.write();
        book.close();
        return true;
  } catch (Exception e) {
        System.out.println("异常信息："+e.getMessage());
        e.printStackTrace();
        return false;
  }
}
```

秘笈心法

在实际应用中，根据客户需求可能会在导出工作表并打印时，需要设置打印纸的页眉、页脚以及纸张大小等属性。所以，应该了解如何通过 JXL 组件来设置 Excel 工作表的打印属性。

实例 551　设置 Excel 工作表详细的打印属性

光盘位置：光盘\MR\21\551

高级

实用指数：★★★

实例说明

在实例 550 中，介绍了如何设置 Excel 工作表的打印属性，但是设置的打印属性只有页眉、页脚以及打印纸这几个，而实际情况可能需要设置更多的打印属性，本实例将介绍如何设置 Excel 工作表其他复杂的打印属性。运行本实例，导出所有员工信息到 Excel 工作表之后，设置 Excel 工作表复杂的打印属性。如图 21.21 所示为在打印预览中的打印效果。

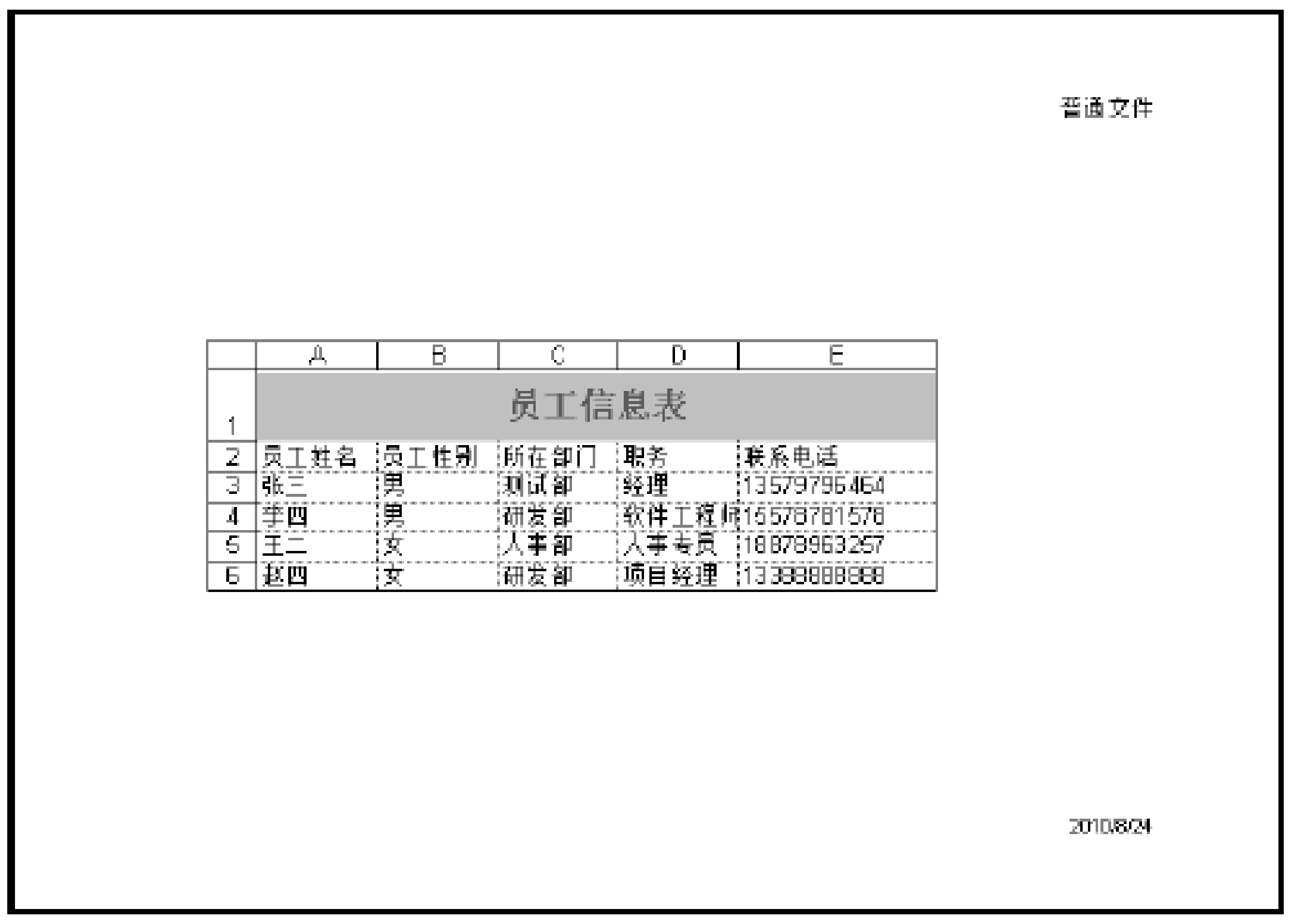


图 21.21　设置 Excel 工作表详细的打印属性

关键技术

本实例在实现时不再应用 WritableSheet 类的几个方法设置打印属性，而是应用 jxl.SheetSettings 类中的方法来进行详细设置。SheetSettings 类主要的用途就是设置和获取 Excel 工作表的打印属性。SheetSettings 类的实例由 jxl.write.WritableSheet 对象的 getSettings()方法获取。SheetSettings 类常用的方法及说明如表 21.2 所示。

表 21.2　SheetSettings 类的常用方法及说明

方　　法	说　　明
setBottomMargin(double m)	设置距离页面底部的间距
setDefaultColumnWidth(int w)	设置默认列宽
setDefaultRowHeight(int h)	设置默认行高
setFitHeight(int fh)	设置页高
setFitWidth(int fw)	设置页宽
setFooter(HeaderFooter f)	设置页脚
setFooterMargin(double d)	设置页脚距页面底端的间距
setHeader(HeaderFooter h)	设置页眉
setHeaderMargin(double d)	设置页眉距页面顶端的间距
setHorizontalCentre(boolean horizCentre)	设置是否水平居中
setHorizontalPrintResolution(int hpw)	设置水平打印分辨率
setLeftMargin(double m)	设置距离页面左侧的间距
setNormalMarginfication(int f)	设置正常分辨率
setOrientation(PageOrientation po)	设置打印方向。横向或纵向
setPageStart(int ps)	设置打印起始页

续表

方　法	说　明
setPaperSize(PaperSize ps)	设置在打印工作表时，打印纸的大小
setPrintArea(int firstCol,int firstRow,int lastCol,int lastRow)	设置打印区域
setPrintTitles(int firstRow,int lastRow,int firstCol,int lastCol)	设置打印标题
setPrintGridLines(boolean b)	设置打印时是否带有网格
setPrintHeaders(boolean b)	设置是否打印页眉
setRightMargin(double m)	设置距离页面右侧的边距
setScaleFactor(int sf)	设置缩放比例
setTopMargin(double m)	设置距离页面顶端的边距
setVerticalCentre(boolean vertCentre)	设置打印内容是否垂直居中

说明： 表 21.2 列出的方法只是 SheetSettings 类的常用方法，其他方法的详细说明请参见 JXL API 文档。

设计过程

（1）创建用于操作 Excel 的工具类 ExcelOperationUtil，在该类中编写 Excel 工作表的打印设置的方法 setSheetPrintStyle()，参数 sheet 表示要设置打印属性的工作表对象，具体代码如下：

```
public WritableSheet setSheetPrintStyle(WritableSheet sheet){
	SheetSettings settings = sheet.getSettings();	//获取工作表的打印属性对象
	settings.setHorizontalCentre(true);	//设置打印内容水平居中
	settings.setVerticalCentre(true);	//设置打印内容垂直居中
	HeaderFooter header = new HeaderFooter();	//创建用于设置页眉页脚的对象
	header.getRight().append("普通文件");	//设置页眉右侧的内容
	header.getRight().setFontSize(8);	//设置页眉右侧内容的字号
	settings.setHeader(header);	//设置页眉
	settings.setFitHeight(5) ;	//设置页高
	settings.setFitWidth(5);	//设置页宽
	HeaderFooter footer = new HeaderFooter();	//创建用于设置页眉页脚的对象
	footer.getRight().setFontSize(8);	//设置页脚右侧字体的字号
	footer.getRight().appendDate();	//设置在页脚右侧添加日期
	settings.setFooter(footer);	//设置页脚
	settings.setFooterMargin(0.5);	//设置页脚与页面底端的距离
	settings.setHeaderMargin(0.5);	//设置页眉与页面顶端的距离
	settings.setOrientation(PageOrientation.PORTRAIT);//设置纵向打印
	settings.setPaperSize(PaperSize.A4) ;	//设置打印纸大小
	settings.setPrintGridLines(true);	//设置打印网格线
	settings.setPrintHeaders(true);	//打印页眉
	settings.setScaleFactor(80) ;	//缩放比例
	return sheet;
}
```

（2）编写用于向 Excel 工作表中添加员工信息的方法 readDataToExcelFile()，参数 list 封装了从数据库表中读取数据的 JavaBean 对象 Employee。在该方法中，导入员工数据的同时，调用步骤（1）的方法 setSheetPrintStyle()设置工作表详细的打印属性，关键代码如下：

```
public boolean readDataToExcelFile(List<Employee> list){
 try {
	WritableWorkbook book = Workbook.createWorkbook(new File("E:\\test\\员工信息.xls"));
	WritableSheet sheet = book.createSheet("员工信息", 0);
	sheet = this.setSheetPrintStyle(sheet);	//调用打印设置的方法，设置打印属性
	……	//此处省略了导出员工信息的代码
	book.write();
	book.close();
	return true;
 } catch (Exception e) {
	e.printStackTrace();
	return false;
 }
}
```

秘笈心法

在实际应用中，根据客户需求可能会在打印工作表时，需要设置详细的打印属性。所以，应该了解如何通过 JXL 组件的 jxl.SheetSettings 类来设置 Excel 工作表详细的打印属性。

21.2　应用 POI 组件操作 Excel

POI 组件是 Apache 组件的一个开源项目，其开发目的是让 Java 语言可以对 Microsoft 的 Office 系列办公软件进行读/写操作。目前 POI 组件已经实现了对 MS Excel、MS Word 及 MS PowerPoint 等格式的文件进行操作的功能。该组件的源代码文件，可以访问 Apache 组件的 POI 组件专栏 http://poi.apache.org/index.html 网站上进行获取。目前该组件的最新版本为 POI 3.7 beta2，该版本为测试版本，可能存在一些不稳定的问题。因此，建议下载 POI 3.6 版本。接下来将介绍如何应用 POI 组件操作 Excel。

实例 552	创建 Excel 文档 光盘位置：光盘\MR\21\552	初级 实用指数：★★★

实例说明

本实例将介绍如何应用 POI 组件创建 Excel 文档。运行本实例，如图 21.22 所示，输入 Excel 文档的名称和保存路径，单击“创建”按钮后，将在指定磁盘路径中创建一个 Excel 文档。

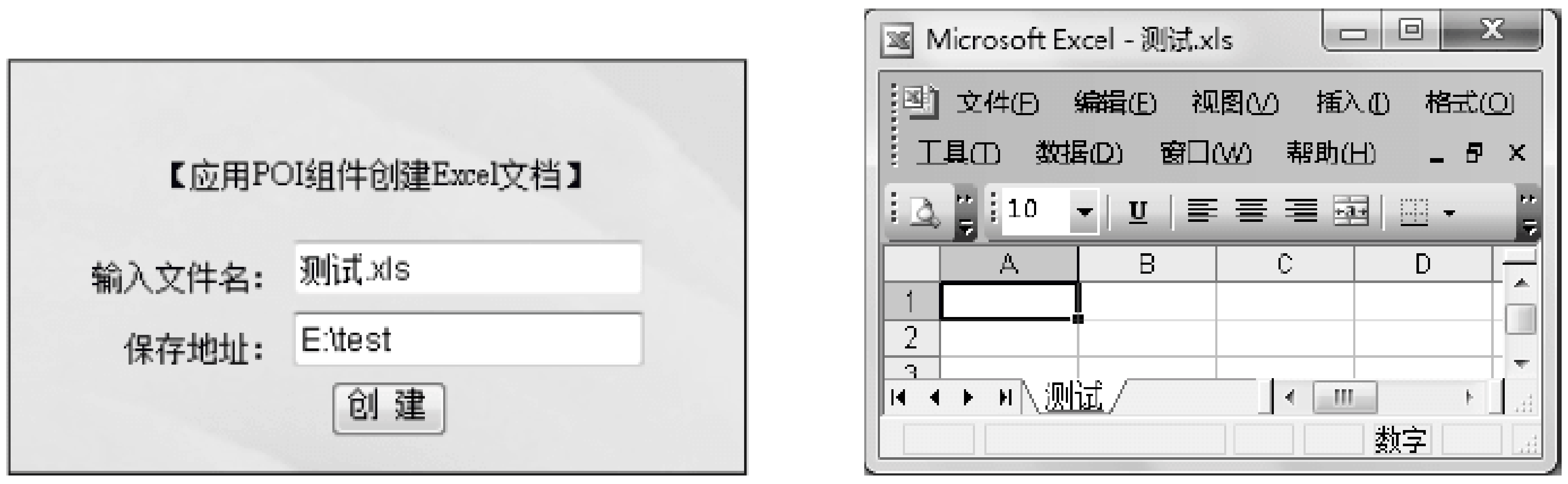


图 21.22　创建 Excel 文档

关键技术

应用 POI 组件创建 Excel 文档，主要应用的是 org.apache.poi.hssf.usermodel.HSSFWorkbook 类和 org.apache.poi.hssf.usermodel.HSSFSheet 类。HSSFWorkbook 表示 Excel 工作簿，也是最为重要的类，因为只有创建 HSSFWorkbook 对象之后，才可以从工作簿中获取工作表对象。HSSFWorkbook 类包含以下常用方法：

（1）createSheet()方法

该方法用于创建 Excel 工作表，语法结构如下：

```
public HSSFSheet createSheet()
```

返回值类型为 HSSFSheet，表示工作表对象。

（2）setSheetName()方法

该方法用于设置 Excel 工作表的名称，语法结构如下：

```
public void setSheetName(int sheetIx,java.lang.String name)
```

参数说明

❶ sheetIx：指定需要修改名称的工作表的索引。索引值从 0 开始。

❷ name：设置工作表的名称。

（3）setSelectedTab()方法

该方法用于设置 Excel 文件中哪一个工作表为选择状态，其语法结构如下：

```
public void setSelectedTab(int index)
```

参数说明

index：指定需要设置选择状态的工作表索引。索引值从 0 开始。

（4）setSheetHidden()方法

该方法用于设置指定的工作表是否隐藏，其语法结构如下：

```
public void setSheetHidden(int sheetIx,boolean hidden)
```

❶ sheetIx：指定需要设置是否隐藏的工作表的索引。索引值从 0 开始。

❷ hidden：设置是否隐藏。取值为 true（隐藏）或 false（不隐藏）。

（5）write()方法

该方法用于将缓存中创建的 Excel 文件对象以流的方式写入文件中，其语法结构如下：

```
public void write(java.io.OutputStream stream) throws java.io.IOException
```

参数说明

stream：文件输出流对象。在将数据写入文件之前，需要创建此文件输出流的对象。

说明： HSSFWorkbook 类中还有很多方法，此处不再进行一一介绍。这些方法的使用说明可查看 POI 组件的 API，API 文档可以在下载的 POI 组件包中找到，也可以到 Apache 官网的 POI 组件指定位置进行查看。在后续的针对不同操作的实例中，也会对这些方法进行详细介绍。

设计过程

创建用于操作 Excel 的工具类 ExcelOperationUtil，在该类中编写创建 Excel 文档的方法 CreateExcelFile()，参数 filePath 表示文件的保存路径，参数 fileName 表示文件的名称，关键代码如下：

```
public boolean CreateExcelFile(String filePath,String fileName){
 try{
        HSSFWorkbook workbook = new HSSFWorkbook();             //创建 Excel 工作簿对象
        HSSFSheet sheet = workbook.createSheet();               //在工作簿中创建工作表对象
        workbook.setSheetName(0, "测试");                        //设置工作表的名称
        HSSFRow row = sheet.createRow(0);                       //在工作表中创建行对象
        HSSFCell cell = row.createCell(0,Cell.CELL_TYPE_STRING);//在行中创建单元格对象
        cell.setCellValue("这是我的第一个 Excel 文档！");          //设置单元格内容
        File xlsFile = new File(filePath,fileName);
        FileOutputStream fos = new FileOutputStream(xlsFile);   //创建文件输出流对象
        workbook.write(fos);                                    //将文档对象写入文件输出流
        fos.close();                                            //关闭文件输出流
        return true;
 } catch (Exception e) {
        e.printStackTrace();
        return false;
 }
}
```

秘笈心法

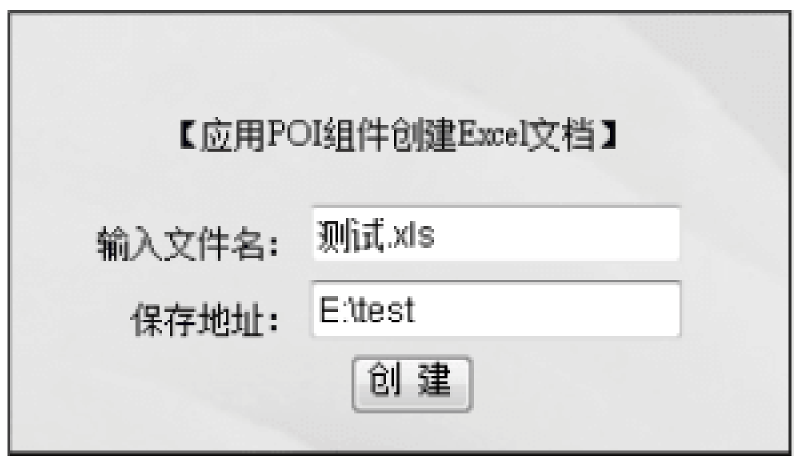
在创建 HSSFWorkbook 对象之后，不要忘记调用其 write()方法将所有对 Excel 文档的设置输出到文件中去，并且及时调用 close()方法关闭文件输出流，释放系统资源。

实例 553	在 Excel 工作表中创建单元格 光盘位置：光盘\MR\21\553	初级 实用指数：★★★

实例说明

本实例将介绍如何应用 POI 组件，创建 Excel 工作表中的单元格。运行本实例，如图 21.23 所示，输入 Excel

文档的名称和保存路径，单击“创建”按钮后，将在指定磁盘路径中创建一个 Excel 文档，并向 Excel 工作表单元格中输入测试信息。



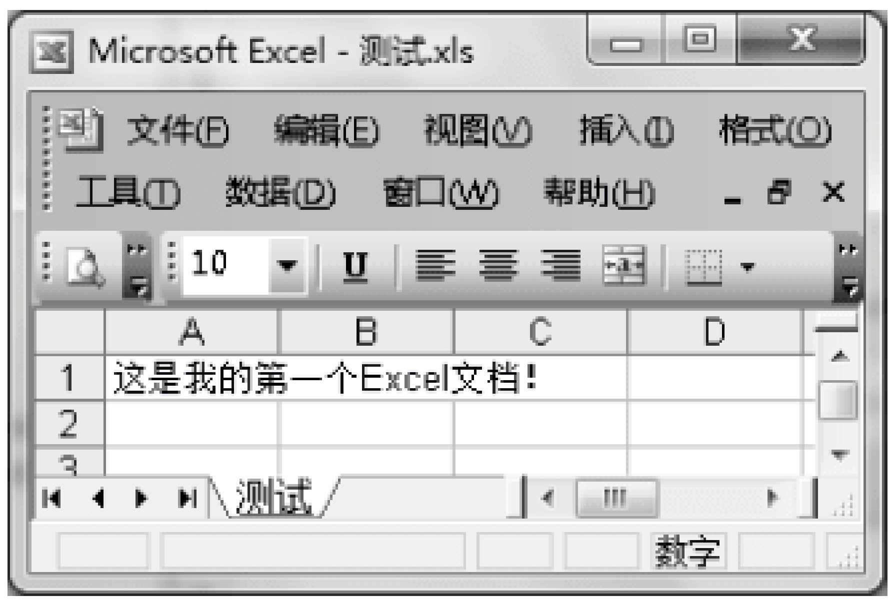


图 21.23　创建 Excel 单元格

关键技术

创建 Excel 单元格之前，首先需要应用 HSSFWorkbook 对象创建 Excel 工作簿，然后应用 HSSFWorkbook 对象的 createSheet()方法创建工作表对象 HSSFSheet，接下来由 HSSFSheet 对象的 createRow()方法创建一个表示 Excel 工作表中行的对象 HSSFRow，最后再由行对象 HSSFRow 的 createCell()方法来创建单元格对象 HSSFCell。下面介绍如何创建行对象 HSSFRow 和单元格对象 HSSFCell。

（1）createRow()方法

该方法是 HSSFSheet 类中的方法，用于创建 Excel 工作表的行对象，语法结构如下：

```
public HSSFRow createRow(int rownum)
```

参数说明

rownum：指定 Excel 工作表的行索引。索引值从 0 开始。

返回值类型为 HSSFRow，表示一个 Excel 工作表的行对象。

（2）createCell()方法

创建行对象 HSSFRow 之后，调用该对象的 createCell()方法创建单元格。createCell()方法的语法结构如下：

```
public HSSFCell createCell(int columnIndex,int type)
```

参数说明

❶ columnIndex：指定行中的单元格索引。索引值从 0 开始，即行中的第一个单元格。

❷ type：可选参数，指定该单元格的内容类型。类型值由 org.apache.poi.ss.usermodel.Cell 接口指定，例如 Cell.CELL_TYPE_STRING（字符串类型）、Cell.CELL_TYPE_BOOLEAN（布尔类型）、Cell.CELL_TYPE_NUMERIC（数字类型）等。

返回值类型为 HSSFCell，表示一个 Excel 工作表的单元格对象。

设计过程

创建用于操作 Excel 的工具类 ExcelOperationUtil，在该类中编写创建 Excel 文档的方法 CreateExcelFile()，参数 filePath 表示文件的保存路径，参数 fileName 表示文件的名称。在 CreateExcelFile()方法中，创建 Excel 工作表之后，在该工作表中创建单元格，关键代码如下：

```
public boolean CreateExcelFile(String filePath,String fileName){
try{
        HSSFWorkbook workbook = new HSSFWorkbook();          //创建 Excel 工作簿对象
        HSSFSheet sheet = workbook.createSheet();            //在工作簿中创建工作表对象
        workbook.setSheetName(0, "测试");                     //设置工作表的名称
        HSSFRow row = sheet.createRow(0);                    //在工作表中创建行对象
        HSSFCell cell = row.createCell(0,Cell.CELL_TYPE_STRING);//在行中创建单元格对象
        cell.setCellValue("这是我的第一个 Excel 文档！ ");      //设置单元格内容
        File xlsFile = new File(filePath,fileName);
        FileOutputStream fos = new FileOutputStream(xlsFile);
        workbook.write(fos);                                 //将文档对象写入文件输出流
        fos.close();
```

```
            return true;
    } catch (Exception e) {
            e.printStackTrace();
            return false;
    }
}
```

秘笈心法

在一个 Excel 文档中，可以将其内容由大到小分为工作簿、工作表、行、单元格等几个元素，POI 组件正是根据这一结构特点来进行处理的。例如，工作表对象 HSSFSheet 是由工作簿 HSSFWorkbook 对象创建的，而工作表中的行对象 HSSFRow 是由工作表对象 HSSFSheet 创建的，单元格对象 HSSFCell 是由行对象 HSSFRow 创建的。

实例 554 向 Excel 单元格中添加不同类型的数据

光盘位置：光盘\MR\21\554

初级

实用指数：★★★

实例说明

本实例将介绍如何应用 POI 组件，向 Excel 工作表单元格中添加不同类型的数据。运行本实例，如图 21.24 所示，输入 Excel 文档的名称和保存路径，单击“创建”按钮后，将在指定磁盘路径中创建一个 Excel 文档，并向 Excel 工作表单元格中添加数字类型的数据、日期时间类型的数据、布尔类型的数据和字符串类型的数据等。



图 21.24　向 Excel 工作表的单元格中添加不同类型的数据

关键技术

向 Excel 工作表的单元格中添加不同类型的数据，主要应用的是单元格 HSSFCell 对象的 setCellValue()方法，该方法包含了用于添加多种不同类型数据的重载方法。下面对这些方法进行简单介绍。

- 添加数字类型的数据

通过 HSSFCell 对象的 setCellValue()方法，可以添加数字类型的数据，语法结构如下：

```
public void setCellValue(double value)
```

参数说明

value：为单元格设置一个 double 类型的值。

- 添加日期类型的数据

通过 HSSFCell 对象的 setCellValue()方法，同样可以添加日期类型的数据，语法结构如下：

```
public void setCellValue(Date date)
```

参数说明

date：设置单元格的数据类型为 java.util.Date。也可以应用 java.util.Calendar 类型的对象作为该方法的参数。

- 添加布尔值类型的数据

通过 HSSFCell 对象的 setCellValue()方法，也可以添加布尔类型的数据，语法结构如下：

```
public void setCellValue(boolean value)
```

参数说明

value：设置单元格数据类型为 boolean 类型。

❑ 添加字符串类型的数据

通过 HSSFCell 对象的 setCellValue()方法，最常用的就是添加字符串类型的数据，语法结构如下：

```
public void setCellValue(String value)
```

参数说明

value：设置单元格数据类型为字符串类型。

设计过程

创建用于操作 Excel 的工具类 ExcelOperationUtil，在该类中编写创建 Excel 文档的方法 CreateExcelFile()，然后在该方法中为 Excel 工作表添加多个单元格，并在这些单元格中添加不同类型的数据，关键代码如下：

```
public boolean CreateExcelFile(String filePath,String fileName){
try{
        HSSFWorkbook workbook = new HSSFWorkbook();     //创建 Excel 工作簿对象
        HSSFSheet sheet = workbook.createSheet();       //在工作簿中创建工作表对象
        workbook.setSheetName(0, "测试");               //设置工作表的名称
        HSSFRow row = sheet.createRow(0);               //在工作表中创建第 1 行对象
        HSSFCell label_num = row.createCell(0);         //第 1 行的第 1 个单元格
        label_num.setCellValue("数字类型");             //添加字符串
        HSSFCell label_date = row.createCell(1);        //第 1 行的第 2 个单元格
        label_date.setCellValue("日期时间类型");        //添加字符串
        HSSFCell label_bool = row.createCell(2);        //第 1 行的第 3 个单元格
        label_bool.setCellValue("布尔类型");            //添加字符串
        HSSFRow row2 = sheet.createRow(1);              //在工作表中创建第 2 行对象
        HSSFCell num_cell = row2.createCell(0);         //第 2 行的第 1 个单元格
        num_cell.setCellValue(3.1415926);               //添加数字
        HSSFCell date_cell = row2.createCell(1);        //第 2 行的第 2 个单元格
        date_cell.setCellValue(Calendar.getInstance()); //添加日期时间
        HSSFCell bool_cell = row2.createCell(2);        //第 2 行的第 3 个单元格
        bool_cell.setCellValue(false);                  //添加布尔值
        File xlsFile = new File(filePath,fileName);
        FileOutputStream fos = new FileOutputStream(xlsFile);
        workbook.write(fos);                            //将文档对象写入文件输出流
        fos.close();
        return true;
} catch (Exception e) {
        e.printStackTrace();
        return false;
}
}
```

秘笈心法

在实际应用中，经常需要在 Excel 工作表中添加不同类型的数据，因此，应该掌握应用 POI 组件向 Excel 工作表单元格中添加不同类型数据的方法。

实例 555　创建指定格式的单元格

光盘位置：光盘\MR\21\555

初级　实用指数：★★★

实例说明

在实例 554 中不难发现，添加日期时间类型的数据之后，显示的是一串数字，而不是平常所看到的日期时间，这是由于只是添加了一个 Calendar 类型或者 Date 类型的对象，想要将日期时间对象显示为正常的日期时间，还需要对单元格进行格式化，也就是设置单元格数据的格式。本实例将介绍如何应用 POI 组件，设置单元格数据的格式。运行本实例，如图 21.25 所示，输入 Excel 文档的名称和保存路径，单击“创建”按钮后，将在指定磁盘路径中创建一个 Excel 文档，并设置指定单元格的格式为日期时间。

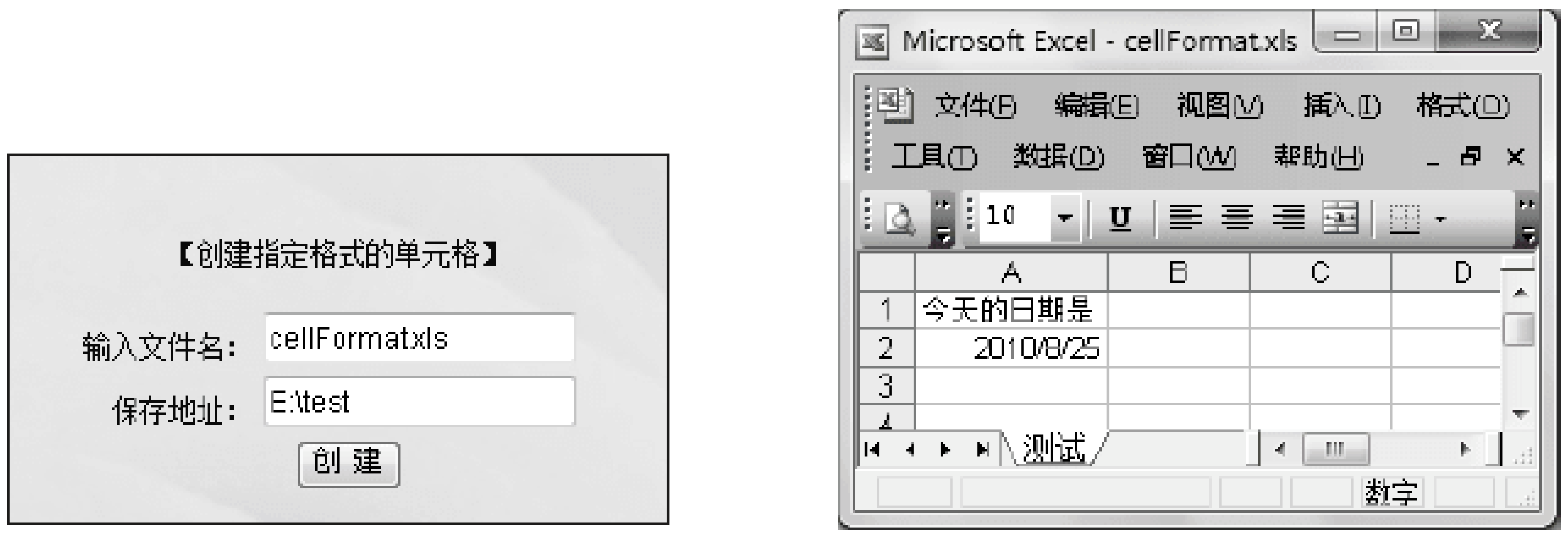


图 21.25　创建指定格式的单元格

关键技术

设置单元格的数据格式，主要应用 org.apache.poi.hssf.usermodel.HSSFCellStyle 类的 setDataFormat()方法，HSSFCellStyle 类主要用于设置单元格的显示样式。在创建一个单元格对象 HSSFCell 之后，可以调用它的 setCellStyle()方法为指定单元格设置样式，setCellStyle()方法的参数也就是 HSSFCellStyle 类型的对象。针对本实例，下面介绍创建指定格式的单元格的步骤。

（1）创建单元格样式对象 HSSFCellStyle。HSSFCellStyle 对象是由工作簿对象 HSSFWorkbook 创建的，其语法结构如下：

```
HSSFWorkbook workbook = new HSSFWorkbook();
HSSFCellStyle cellStyle = workbook.createCellStyle();
```

（2）设置单元格的数据格式。设置单元格的数据格式主要应用的是 HSSFDataFormat 类的静态方法 getBuiltinFormat()，通过设置该方法不同格式的参数来设置单元格数据的不同格式。getBuiltinFormat()方法的语法结构如下：

```
public static short getBuiltinFormat(java.lang.String format)
```

参数说明

format：指定字符串类型的格式化参数。通过指定不同类型的格式化参数，可以实现对单元格的数据进行不同格式化操作，如 d-mmm、mmm-yy、m/d/yy、m/d/yy h:mm、0%、0.00%、0.00 等。有关这些参数的详细设置请参见 POI 组件的 API 文档，此处不进行详细说明。

（3）设置完单元格的数据格式之后，需要调用 HSSFCellStyle 对象的 setDataFormat()方法，将格式应用于单元格。setDataFormat()方法的语法格式如下：

```
public void setDataFormat(short fmt)
```

参数说明

fmt：指定单元格的数据格式。这个 short 类型的格式参数值是通过 HSSFDataFormat 类的 getBuiltinFormat()方法返回的。

设计过程

创建用于操作 Excel 的工具类 ExcelOperationUtil，在该类中编写创建 Excel 文档的方法 CreateExcelFile()，然后在方法中为 Excel 工作表添加一个单元格，并设置单元格的格式为日期时间，关键代码如下：

```
public boolean CreateExcelFile(String filePath,String fileName){
try{
        HSSFWorkbook workbook = new HSSFWorkbook();                    //创建 Excel 工作簿对象
        HSSFSheet sheet = workbook.createSheet();                      //在工作簿中创建工作表对象
        workbook.setSheetName(0, "测试");                              //设置工作表的名称
        HSSFRow row1 = sheet.createRow(0);                             //在工作表中创建行对象
        HSSFCell Label_cell = row1.createCell(0);                      //在行中创建单元格对象
        Label_cell.setCellValue("今天的日期是");
        HSSFRow row2 = sheet.createRow(1);                             //在工作表中创建行对象
        HSSFCell date_cell = row2.createCell(0);                       //在行中创建单元格对象
        HSSFCellStyle cellStyle = workbook.createCellStyle();          //创建单元格样式对象
        cellStyle.setDataFormat(HSSFDataFormat.getBuiltinFormat("m/d/yy"));  //设置单元格的数据格式
        date_cell.setCellValue(Calendar.getInstance());
        date_cell.setCellStyle(cellStyle);                             //将单元格样式应用于单元格
        File xlsFile = new File(filePath,fileName);
        FileOutputStream fos = new FileOutputStream(xlsFile);
```

```
        workbook.write(fos);                                    //将文档对象写入文件输出流
        fos.close();
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
}
```

秘笈心法

在实际应用中，根据客户需求可能需要在 Excel 工作表中添加各种格式的数据，这就需要通过 HSSFDataFormat 类来对单元格的数据进行格式化，因此需要了解 HSSFDataFormat 类的常用格式化的参数。

实例 556　设置单元格内容的水平对齐方式

光盘位置：光盘\MR\21\556

初级

实用指数：★★★

实例说明

本实例将介绍如何应用 POI 组件，设置 Excel 工作表的单元格内容的水平对齐方式。运行本实例，如图 21.26 所示，输入 Excel 文档的名称和保存路径，单击"创建"按钮后，将在指定磁盘路径中创建一个 Excel 文档，并设置单元格内容为水平居中对齐。

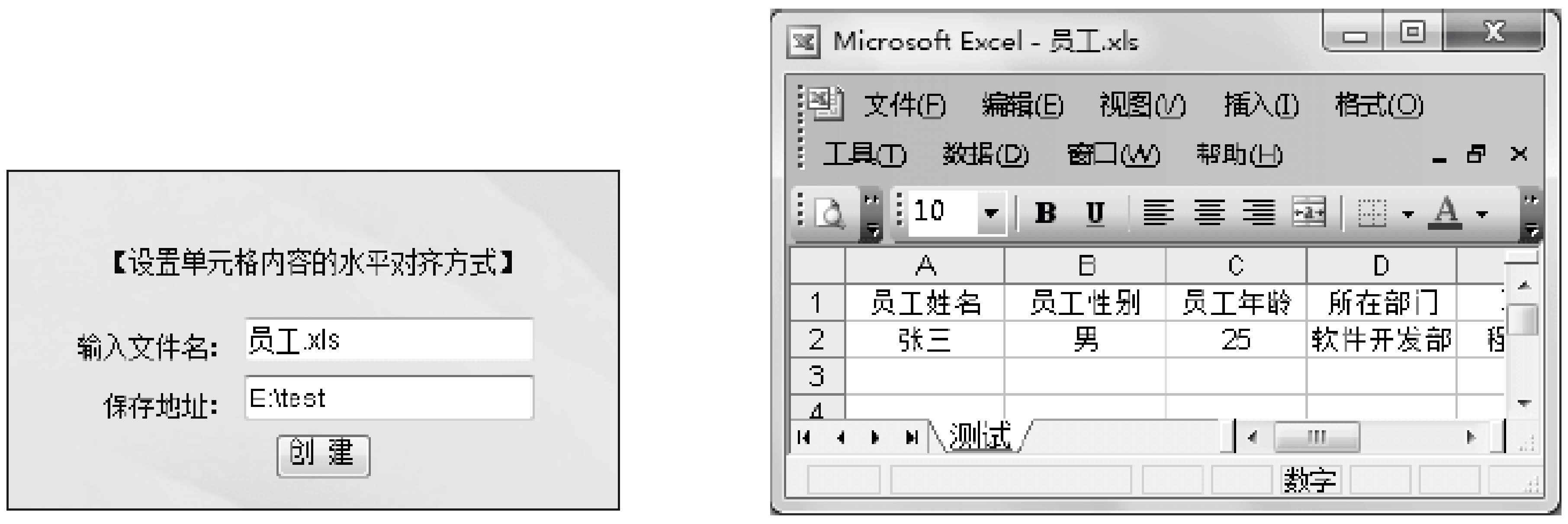


图 21.26　设置单元格内容的水平对齐方式

关键技术

设置单元格内容的水平对齐方式，主要应用 HSSFCellStyle 类的 setAlignment()方法，该方法的语法结构如下：

```
public void setAlignment(short align)
```

参数说明

align：short 类型的参数，用于指定单元格的水平对齐方式。参数值是在 org.apache.poi.ss.usermodel.CellStyle 接口中定义的一组 short 类型的常量，由于 HSSFCellStyle 类实现了 CellStyle 接口，所以这些常量值也可以由 HSSFCellStyle 类来调用。取值包括 ALIGN_CENTER（居中对齐）、ALIGN_CENTER_SELECTION（跨列居中）、ALIGN_FILL（填充对齐）、ALIGN_GENERAL（常规对齐）、ALIGN_JUSTIFY（两端对齐）、ALIGN_LEFT（左对齐）、ALIGN_RIGHT（右对齐）。

设计过程

（1）创建用于操作 Excel 的工具类 ExcelOperationUtil，编写设置对齐方式的方法 createStyle()，参数 wb 表示要设置对齐方式的 Excel 工作簿，参数 align 表示对齐方式的取值，具体代码如下：

```
private static HSSFCellStyle createStyle(HSSFWorkbook wb,short align){
    HSSFCellStyle cellStyle = wb.createCellStyle();
    cellStyle.setAlignment(align);
    return cellStyle;
}
```

（2）编写创建 Excel 文档的方法 CreateExcelFile()，然后在方法中为 Excel 工作表添加表示员工信息的单元格，并调用步骤（1）的方法 createStyle()设置每个单元格的水平对齐方式，关键代码如下：

```
public boolean CreateExcelFile(String filePath,String fileName){
try{
        HSSFWorkbook workbook = new HSSFWorkbook();                              //创建 Excel 工作簿对象
        HSSFSheet sheet = workbook.createSheet();                                //在工作簿中创建工作表对象
        workbook.setSheetName(0, "测试");                                        //设置工作表的名称
        HSSFRow row1 = sheet.createRow(0);                                       //在工作表中创建行对象
        HSSFCell nameCell = row1.createCell(0);                                  //在第 1 行中创建单元格对象
        nameCell.setCellValue("员工姓名");
        nameCell.setCellStyle(createStyle(workbook,HSSFCellStyle.ALIGN_CENTER));    //设置居中
        HSSFCell sexCell = row1.createCell(1);                                   //在行中创建单元格对象
        sexCell.setCellValue("员工性别");
        sexCell.setCellStyle(createStyle(workbook,HSSFCellStyle.ALIGN_CENTER));     //设置居中
        HSSFCell ageCell = row1.createCell(2);                                   //在行中创建单元格对象
        ageCell.setCellValue("员工年龄");
        ageCell.setCellStyle(createStyle(workbook,HSSFCellStyle.ALIGN_CENTER));     //设置居中
        ……//此处省略了添加其他单元格的代码
        HSSFRow row2 = sheet.createRow(1);                                       //在工作表中创建行对象
        HSSFCell nameValue = row2.createCell(0);                                 //在第 2 行中创建单元格对象
        nameValue.setCellValue("张三");
        nameValue.setCellStyle(createStyle(workbook,HSSFCellStyle.ALIGN_CENTER));   //设置居中
        HSSFCell sexValue = row2.createCell(1);                                  //在行中创建单元格对象
        sexValue.setCellValue("男");
        sexValue.setCellStyle(createStyle(workbook,HSSFCellStyle.ALIGN_CENTER));    //设置居中
        HSSFCell ageValue = row2.createCell(2);                                  //在行中创建单元格对象
        ageValue.setCellValue(25);
        ageValue.setCellStyle(createStyle(workbook,HSSFCellStyle.ALIGN_CENTER));    //设置居中
        ……//此处省略了添加其他单元格的代码
        File xlsFile = new File(filePath,fileName);
        FileOutputStream fos = new FileOutputStream(xlsFile);
        workbook.write(fos);                                                     //将文档对象写入文件输出流
        fos.close();
        return true;
} catch (Exception e) {
        e.printStackTrace();
        return false;
}
}
```

秘笈心法

在实际应用中，经常需要设置 Excel 工作表单元格内容的水平对齐方式，所以，掌握应用 POI 组件设置单元格的对齐方式是很有必要的。

实例 557	设置单元格内容的垂直对齐方式 光盘位置：光盘\MR\21\557	初级 实用指数：★★★

实例说明

在实例 556 中介绍了如何设置单元格的水平对齐方式，本实例将介绍如何应用 POI 组件，设置 Excel 工作表的单元格内容的垂直对齐方式。运行本实例，如图 21.27 所示，输入 Excel 文档的名称和保存路径，单击“创建”按钮后，将在指定磁盘路径中创建一个 Excel 文档，并设置单元格内容为垂直居中对齐。

关键技术

设置单元格内容的垂直对齐方式，主要应用 HSSFCellStyle 类的 setVerticalAlignment()方法，该方法的语法结构如下：

```
public void setVerticalAlignment(short align)
```

参数说明

align：short 类型的参数，用于指定单元格的垂直对齐方式。参数值是在 CellStyle 接口中定义的一组 short

类型的常量，由于 HSSFCellStyle 类实现了 CellStyle 接口，所以这些常量值也可以由 HSSFCellStyle 类来调用。取值包括 VERTICAL_BOTTOM（底端对齐）、VERTICAL_CENTER（居中对齐）、VERTICAL_JUSTIFY（两端对齐）、VERTICAL_TOP（顶端对齐）。



图 21.27　设置单元格内容的垂直对齐方式

设计过程

（1）创建用于操作 Excel 的工具类 ExcelOperationUtil，编写设置对齐方式的方法 alignStyle()，参数 wb 表示要设置对齐方式的 Excel 工作簿，参数 align 表示水平对齐方式，参数 v_align 表示垂直对齐方式，具体代码如下：

```
private static HSSFCellStyle alignStyle(HSSFWorkbook wb,short align,short v_align)
{
    HSSFCellStyle cellStyle = wb.createCellStyle();
    cellStyle.setAlignment(align);
    cellStyle.setVerticalAlignment(v_align);
    return cellStyle;
}
```

（2）编写创建 Excel 文档的方法 CreateExcelFile()，然后在方法中为 Excel 工作表添加表示员工信息的单元格，并调用步骤（1）的方法 alignStyle()设置每个单元格水平对齐方式和垂直对齐方式，关键代码如下：

```
public boolean CreateExcelFile(String filePath,String fileName){
try{
        HSSFWorkbook workbook = new HSSFWorkbook();              //创建 Excel 工作簿对象
        HSSFSheet sheet = workbook.createSheet();                //在工作簿中创建工作表对象
        workbook.setSheetName(0, "测试");                         //设置工作表的名称
        HSSFRow row1 = sheet.createRow(0);                       //在工作表中创建行对象
        HSSFCell nameCell = row1.createCell(0);                  //在第 1 行中创建单元格对象
        nameCell.setCellValue("员工姓名");
        nameCell.setCellStyle(alignStyle(workbook,HSSFCellStyle.ALIGN_CENTER,HSSFCellStyle.VERTICAL_CENTER)); //设置单元格对齐方式
        HSSFCell sexCell = row1.createCell(1);                   //在行中创建单元格对象
        sexCell.setCellValue("员工性别");
        sexCell.setCellStyle(alignStyle(workbook,HSSFCellStyle.ALIGN_CENTER,HSSFCellStyle.VERTICAL_CENTER)); //设置单元格对齐方式
        ……//此处省略了添加其他单元格以及设置样式的代码
        HSSFRow row2 = sheet.createRow(1);                       //在工作表中创建行对象
        HSSFCell nameValue = row2.createCell(0);                 //在第 2 行中创建单元格对象
        nameValue.setCellValue("张三");
        nameValue.setCellStyle(alignStyle(workbook,HSSFCellStyle.ALIGN_CENTER,HSSFCellStyle.VERTICAL_CENTER));  //设置单元格对齐方式

        HSSFCell sexValue = row2.createCell(1);                  //在行中创建单元格对象
        sexValue.setCellValue("男");
        sexValue.setCellStyle(alignStyle(workbook,HSSFCellStyle.ALIGN_CENTER,HSSFCellStyle.VERTICAL_CENTER)); //设置单元格对齐方式
        ……//此处省略了添加其他单元格以及设置样式的代码
        File xlsFile = new File(filePath,fileName);
        FileOutputStream fos = new FileOutputStream(xlsFile);
        workbook.write(fos);                                     //将文档对象写入文件输出流
        fos.close();
        return true;
} catch (Exception e) {
        e.printStackTrace();
        return false;
}
}
```

秘笈心法

在实际应用中，有时需要设置 Excel 工作表单元格内容的垂直对齐方式，所以，掌握应用 POI 组件设置单元格的垂直对齐方式也是很有必要的。

实例 558 合并单元格

光盘位置：光盘\MR\21\558

初级

实用指数：★★★

实例说明

本实例将介绍如何应用 POI 组件，对 Excel 工作表的单元格进行合并操作。运行本实例，如图 21.28 所示，输入 Excel 文档的名称和保存路径，单击“创建”按钮后，将在指定磁盘路径中创建一个 Excel 文档，并对指定的单元格进行合并操作。

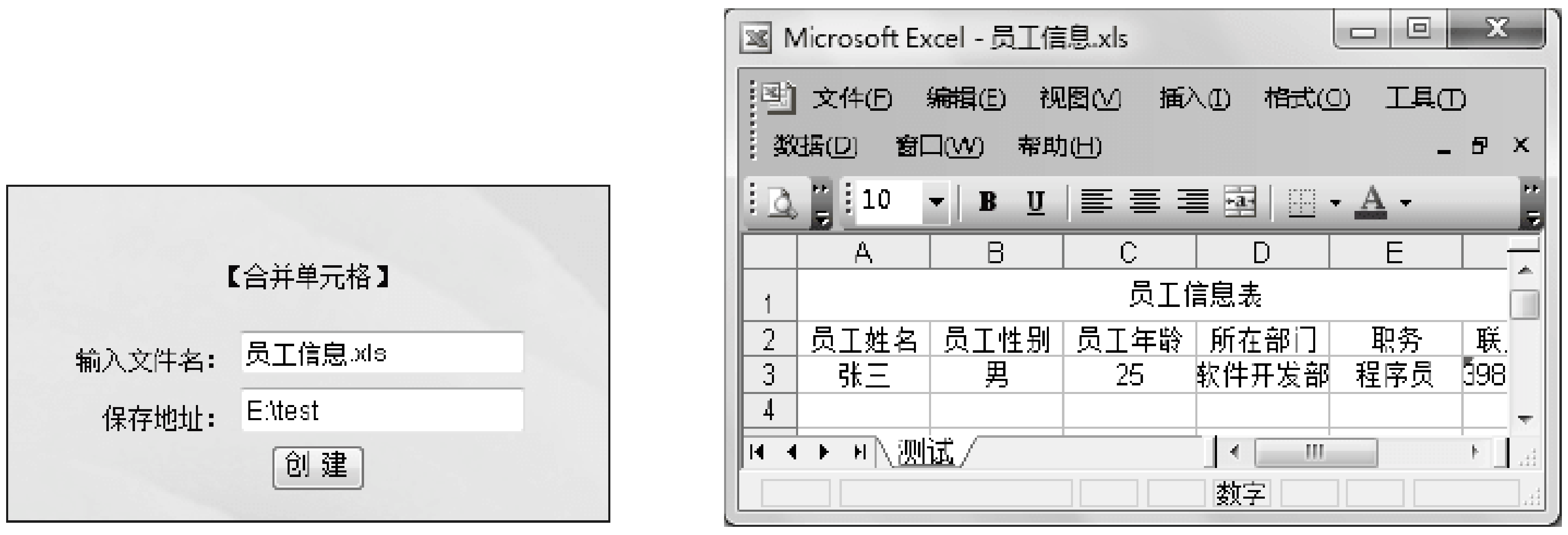


图 21.28　合并单元格

关键技术

对 Excel 工作表的单元格进行合并操作，主要应用的是 HSSFSheet 类的 addMergedRegion()方法，该方法包含一个用于设置合并区域的 org.apache.poi.hssf.util.Region 类型的对象，应用 Region 对象可以对单元格进行合并操作。Region 类的其中一个构造方法的语法结构如下：

```
public Region(int rowFrom,short colFrom,int rowTo,short colTo)
```

参数说明

❶ rowFrom：指定要合并的单元格所在行的起始位置的行索引。索引值从 0 开始。

❷ colFrom：指定要合并的单元格所在列的起始位置的列索引。索引值从 0 开始。

❸ rowTo：指定要合并的单元格所在行的结束位置的行索引。索引值从 0 开始。

❹ colTo：指定要合并的单元格所在列的结束位置的列索引。索引值从 0 开始。

提示： *在创建 Region 对象时，在构造方法中可以不用指定参数，也就是应用它的无参的构造方法，然后通过该对象的 setXXX()方法也可以设置合并单元格的区域的值。*

设计过程

（1）创建用于操作 Excel 的工具类 ExcelOperationUtil，编写设置对齐方式的方法 alignStyle()，参数 wb 表示要设置对齐方式的 Excel 工作簿，参数 align 表示水平对齐方式，参数 v_align 表示垂直对齐方式，具体代码如下：

```
private static HSSFCellStyle alignStyle(HSSFWorkbook wb,short align,short v_align)
{
    HSSFCellStyle cellStyle = wb.createCellStyle();
    cellStyle.setAlignment(align);
    cellStyle.setVerticalAlignment(v_align);
    return cellStyle;
}
```

（2）编写创建 Excel 文档的方法 CreateExcelFile()，然后在方法中为 Excel 工作表添加表示员工信息的单元格，并且合并第一行指定的多个单元格为一个单元格，关键代码如下：

```
public boolean CreateExcelFile(String filePath,String fileName){
 try{
        HSSFWorkbook workbook = new HSSFWorkbook();           //创建 Excel 工作簿对象
        HSSFSheet sheet = workbook.createSheet();              //在工作簿中创建工作表对象
        workbook.setSheetName(0, "测试");                       //设置工作表的名称
        HSSFRow row1 = sheet.createRow(0);                     //在工作表中创建行对象
        sheet.addMergedRegion(new Region(0,(short)0,0,(short)5)); //合并第 1 行的第 1～第 5 个单元格
        HSSFCell titleCell = row1.createCell(0);
        titleCell.setCellValue("员工信息表");
        titleCell.setCellStyle(alignStyle(workbook,HSSFCellStyle.ALIGN_CENTER,HSSFCellStyle.VERTICAL_CENTER));
        HSSFRow row2 = sheet.createRow(1);
        HSSFCell nameCell = row2.createCell(0);                //在第 1 行中创建单元格对象
        nameCell.setCellValue("员工姓名");
        ……//此处省略了添加其他单元格的代码
        File xlsFile = new File(filePath,fileName);
        FileOutputStream fos = new FileOutputStream(xlsFile);
        workbook.write(fos);                                   //将文档对象写入文件输出流
        fos.close();
        return true;
 } catch (Exception e) {
        e.printStackTrace();
        return false;
 }
}
```

秘笈心法

在实际应用中，根据实际情况经常需要对 Excel 工作表单元格进行合并，所以，掌握应用 POI 组件合并单元格是很有必要的。

实例 559　**设置单元格的边框样式**　初级

光盘位置：光盘\MR\21\559　实用指数：★★★

实例说明

本实例将介绍如何应用 POI 组件，设置 Excel 工作表的单元格的边框样式。运行本实例，如图 21.29 所示，输入 Excel 文档的名称和保存路径，单击“创建”按钮后，将在指定磁盘路径中创建一个 Excel 文档，并对指定的单元格的边框样式进行设置。

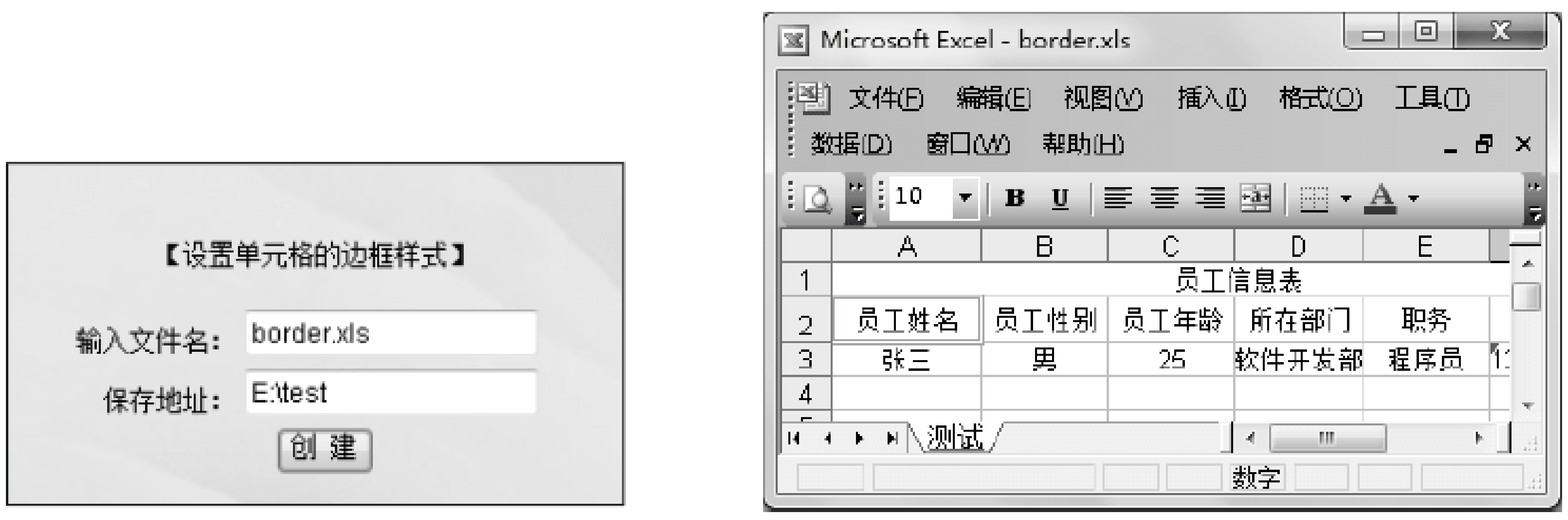


图 21.29　设置单元格的边框样式

关键技术

设置单元格的边框样式，主要应用的是 HSSFCellStyle 类的 setBorderXXX()方法。下面介绍用于设置单元格上边框样式的方法 setBorderTop()。

该方法用于设置单元格的上边框的线条样式，如可以将上边框设置成虚线、实线、双实线等。其语法结构

如下：

```
public void setBorderTop(short border)
```

参数说明

border：指定边框的线条样式。取值为 HSSFCellStyle 类中的静态常量，例如 BORDER_DOUBLE（双实线）、BORDER_THIN（细线）、BORDER_THICK（粗线）等。有关其他详细参数请参见 POI 组件的 API 中的 HSSFCellStyle 类。

说明：在 HSSFCellStyle 类中，用于设置左边框、右边框和下边框的 setBorderXXX()方法与 setBorderTop()方法基本类似，在此不再详细介绍。

HSSFCellStyle 类中还包含用于设置单元格上、下、左、右边框颜色的方法 setXXXBorderColor()。下面介绍用于设置单元格上边框颜色的方法 setTopBorderColor。

该方法用于设置单元格的上边框颜色，其语法结构如下：

```
public void setTopBorderColor(short color)
```

color：指定边框的颜色值。取值为 HSSFColor 类中的静态常量，包括 40 多种常用颜色值，例如 GREEN、BLUE、YELLOW、PINK、GRAY 等。这些颜色是 HSSFColor 类中定义的常量类型的子类，在每个子类中都提供了一个表示本颜色的 short 类型的常量索引 index。因此在为 setTopBorderColor()设置参数时，必须以 HSSFColor.GREEN.index 这种方式进行设置。

设计过程

创建用于操作 Excel 的工具类 ExcelOperationUtil，编写创建 Excel 文档的方法 CreateExcelFile()，然后在方法中为指定的单元格设置边框样式，关键代码如下：

```
public boolean CreateExcelFile(String filePath,String fileName){
try{
        HSSFWorkbook workbook = new HSSFWorkbook();                    //创建 Excel 工作簿对象
        HSSFSheet sheet = workbook.createSheet();                      //在工作簿中创建工作表对象
        workbook.setSheetName(0, "测试");                              //设置工作表的名称
        HSSFRow row1 = sheet.createRow(0);                             //在工作表中创建行对象
        sheet.addMergedRegion(new Region(0,(short)0,0,(short)5));      //合并第 1 行的第 1～5 个单元格
        HSSFCell titleCell = row1.createCell(0);
        titleCell.setCellValue("员工信息表");
        HSSFCellStyle cellStyle = workbook.createCellStyle();
        cellStyle.setAlignment(CellStyle.ALIGN_CENTER);                //设置水平居中
        cellStyle.setVerticalAlignment(CellStyle.VERTICAL_CENTER);     //设置垂直居中
        cellStyle.setBorderTop(HSSFCellStyle.BORDER_DOUBLE);           //设置单元格顶部边框的线条样式
        cellStyle.setBorderLeft(HSSFCellStyle.BORDER_DOUBLE);          //设置单元格左侧边框的线条样式
        cellStyle.setBorderRight(HSSFCellStyle.BORDER_DOUBLE);         //设置单元格右侧边框的线条样式
        cellStyle.setBorderBottom(HSSFCellStyle.BORDER_DOUBLE);        //设置单元格底部边框的线条样式
        cellStyle.setLeftBorderColor(HSSFColor.SKY_BLUE.index);        //设置单元格左侧边框颜色
        cellStyle.setRightBorderColor(HSSFColor.SKY_BLUE.index);       //设置单元格右侧边框颜色
        cellStyle.setBottomBorderColor(HSSFColor.SKY_BLUE.index);      //设置单元格底部边框颜色
        cellStyle.setTopBorderColor(HSSFColor.SKY_BLUE.index);         //设置单元部顶部边框颜色

        HSSFRow row2 = sheet.createRow(1);
        HSSFCell nameCell = row2.createCell(0);                        //在第 1 行中创建单元格对象
        nameCell.setCellValue("员工姓名");
        nameCell.setCellStyle(cellStyle);
        ……//此处省略了添加其他单元格的代码
        File xlsFile = new File(filePath,fileName);
        FileOutputStream fos = new FileOutputStream(xlsFile);
        workbook.write(fos);                                           //将文档对象写入文件输出流
        fos.close();
        return true;
} catch (Exception e) {
        e.printStackTrace();
        return false;
}
}
```

秘笈心法

需要注意的是，如果需要设置边框颜色的单元格为合并后的单元格，那么，再为它设置的边框颜色只对合并之前的第一个单元格起作用，对其他单元格无效。如果希望改变合并单元格的边框颜色，在合并之前就应该设置所有要合并的单元格边框颜色，这样合并之后的单元格边框颜色就改变了。

实例 560　**设置字体样式**　初级

光盘位置：光盘\MR\21\560　实用指数：★★★

实例说明

本实例将介绍如何应用 POI 组件，设置 Excel 工作表的单元格字体样式。运行本实例，如图 21.30 所示，输入 Excel 文档的名称和保存路径，单击“创建”按钮后，将在指定磁盘路径中创建一个 Excel 文档，并设置了指定单元格的字体样式。

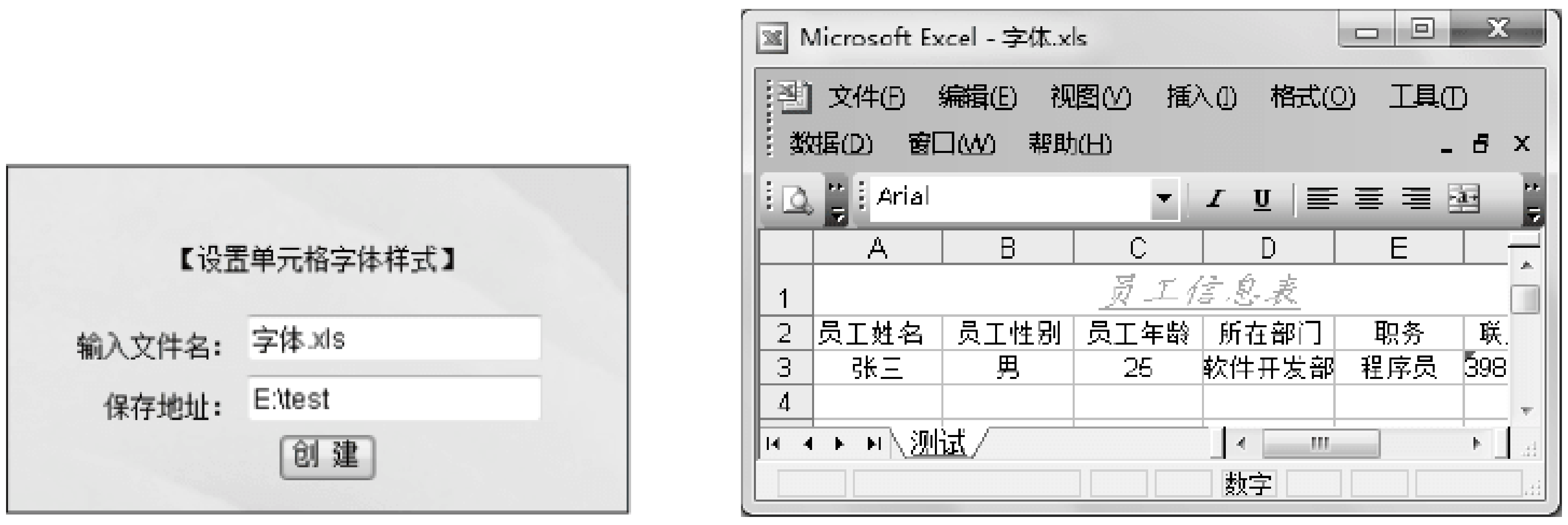


图 21.30　设置单元格的字体样式

关键技术

设置单元格的字体样式，主要应用的是 org.apache.poi.hssf.usermodel.HSSFFont 类，在该类中包含了一些用于设置字体样式的方法。下面介绍 HSSFFont 类中常用的方法。

- ❑ setColor()方法

该方法用于设置字体颜色，其语法结构如下：

```
public void setColor(short color)
```

参数说明

color：指定 short 类型的颜色值。颜色取值由 HSSFColor 类提供。

- ❑ setFontHeightInPoints()方法

该方法用于设置字号，其语法结构如下：

```
public void setFontHeightInPoints(short height)
```

参数说明

height：指定单元格字体的字号。

- ❑ setFontName()方法

该方法用于设置字体名称，其语法结构如下：

```
public void setFontName(String name)
```

参数说明

name：指定单元格字体的名称。可以设置 Excel 文件常用的字体作为该方法的参数，例如 Tahoma、Arial、Times New Roman 等。如果不设置字体名称，默认采用 Arial 类型。

- ❑ setItalic()方法

该方法用于设置字体是否倾斜，其语法结构如下：

```
public void setItalic(boolean italic)
```

参数说明

italic：取值为 true 或 false，true 表示倾斜，false 表示不倾斜。

❑ seUnderline()方法

该方法用于设置字体的下划线，其语法结构如下：

```
public void seUnderline (byte underline)
```

参数说明

underline：指定下划线的样式。取值为 HSSFFont 类提供的静态常量。

说明：HSSFFont 类中还包含了其他几个用于设置字体样式的方法，此处不再进行详细介绍。详细说明请参见 POI 组件的 API 文档。

设计过程

创建用于操作 Excel 的工具类 ExcelOperationUtil，编写创建 Excel 文档的方法 CreateExcelFile()，然后在方法中为指定的单元格设置字体样式，关键代码如下：

```
public boolean CreateExcelFile(String filePath,String fileName){
try{
        HSSFWorkbook workbook = new HSSFWorkbook();             //创建 Excel 工作簿对象
        HSSFSheet sheet = workbook.createSheet();               //在工作簿中创建工作表对象
        workbook.setSheetName(0, "测试");                       //设置工作表的名称
        HSSFRow row1 = sheet.createRow(0);                      //在工作表中创建行对象
        sheet.addMergedRegion(new Region(0,(short)0,0,(short)5)); //合并第 1 行的第 1～5 个单元格
        HSSFFont font = workbook.createFont();                  //创建字体对象
        font.setColor(HSSFColor.SKY_BLUE.index);                //设置字体颜色
        font.setFontHeightInPoints((short)14);                  //设置字号
        font.setFontName("楷体");                               //设置字体样式
        font.setItalic(true);                                   //是否倾斜
        font.setStrikeout(false);                               //是否带有删除线
        font.setUnderline(HSSFFont.U_SINGLE);                   //设置下划线
        HSSFCellStyle cellStyle = workbook.createCellStyle();
        cellStyle.setFont(font);                                //将字体设置添加到样式中
        HSSFCell titleCell = row1.createCell(0);
        titleCell.setCellValue("员工信息表");
        titleCell.setCellStyle(cellStyle);
        ……//此处省略了添加其他单元格的代码
        File xlsFile = new File(filePath,fileName);
        FileOutputStream fos = new FileOutputStream(xlsFile);
        workbook.write(fos);                                    //将文档对象写入文件输出流
        fos.close();
        return true;
} catch (Exception e) {
        e.printStackTrace();
        return false;
}
}
```

秘笈心法

在实际开发应用过程中，有时需要改变 Excel 单元格的字体样式，这就需要应用 POI 组件的 HSSFFont 类进行设置。因此，有必要掌握 HSSFFont 类的用法。

实例 561	向 Excel 文件中插入图片 光盘位置：光盘\MR\21\561	高级 实用指数：★★★

实例说明

本实例将介绍如何应用 POI 组件，向 Excel 工作表中插入图片。运行本实例，如图 21.31 所示，输入 Excel 文档的名称和保存路径，并选中要插入的图片，单击“创建”按钮后，将在指定磁盘路径中创建一个 Excel 文

档，并向 Excel 文件中插入图片。

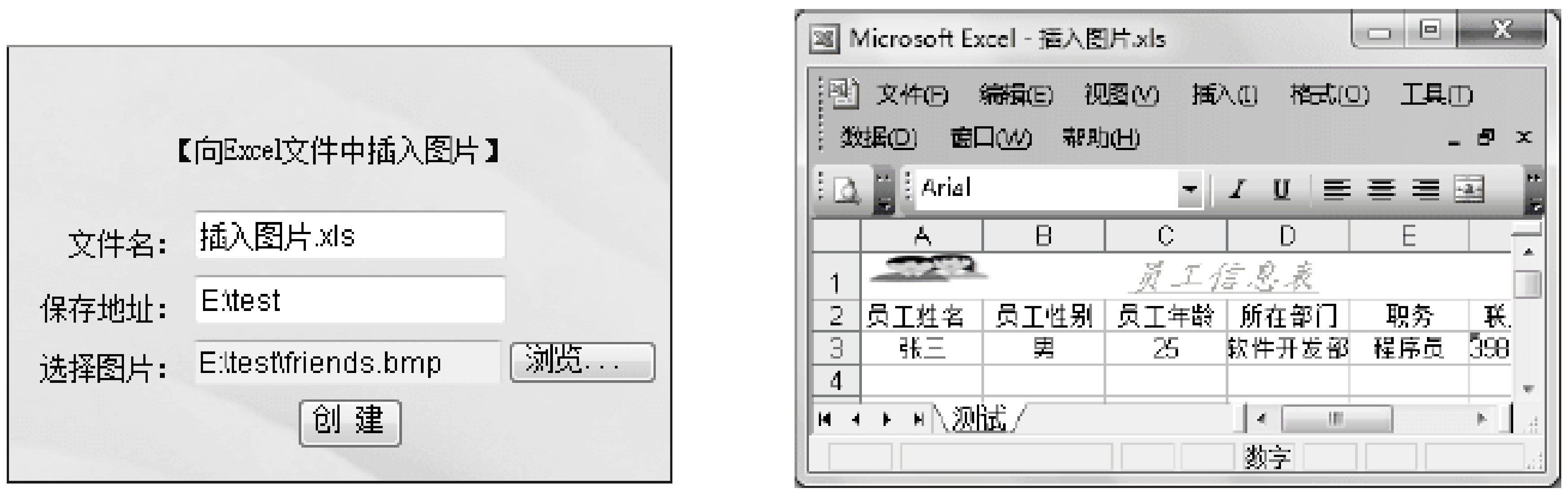


图 21.31　向 Excel 文件中插入图片

关键技术

在应用 POI 组件向 Excel 文件中插入图片之前，应该应用 java.io.FileInputStream 将图片转换为原始字节流数据，然后再应用 POI 组件的相关类进行写入操作，具体步骤如下：

（1）应用 FileInputStream 读取图片的字节流数据，并保存在字节数组中，读取方法代码如下：

```
FileInputStream fis = new FileInputStream(imgPath);
byte [] imgBytes = new byte[fis.available()];//available()方法用于获取图片的字节长度
```

（2）应用 POI 组件中的 org.apache.poi.hssf.usermodel.HSSFClientAnchor 类，在 Excel 工作表中创建一个显示图片的区域。这个类对于向 Excel 文件中插入图片非常重要，因为需要用它来设置图片的显示区域。应用 HSSFClientAnchor 类创建区域时，主要设置区域所在的左上角起始位置坐标和右下角的结束位置坐标，在该类中主要应用 setAnchor()方法来设置区域的坐标。setAnchor()方法的语法结构如下：

```
public void setAnchor(short col1,int row1,int x1,int y1,short col2,int row2,int x2,int y2)
```

参数说明

❶ col1：指定区域所在的起始位置的单元格列索引。

❷ row1：指定区域所在的起始位置的单元格行索引。

❸ x1：指定区域所在的左上角的横坐标。

❹ y1：指定区域所在的左上角的纵坐标。

❺ col2：指定区域所在的结束位置的单元格列索引。

❻ row2：指定区域所在的结束位置的单元格行索引。

❼ x2：指定区域所在的右下角的横坐标。

❽ y2：指定区域所在的右下角的纵坐标。

（3）应用工作簿对象 HSSFWorkbook 中的 addPicture()方法添加图片。调用这个方法后，并不会直接将图片插入 Excel 文件中，而是保存在缓存中，并且返回图片在缓存中的索引值。addPicture()方法的语法结构如下：

```
public int addPicture(byte[] pictureData,int format)
```

参数说明

❶ pictureData：指定图片数据的字节数组。

❷ format：指定保存到 Excel 文件之后的图片类型。取值为 HSSFWorkbook 类中的静态常量，如 PICTURE_TYPE_JPEG、PICTURE_TYPE_PNG、PICTURE_TYPE_WMF 等。

（4）应用 org.apache.poi.hssf.usermodel.HSSFPatriarch 类的 createPicture()方法，将缓存中的图片数据画到 Excel 文件中。HSSFPatriarch 类的 createPicture()方法的语法结构如下：

```
public HSSFPicture createPicture(HSSFClientAnchor anchor,int pictureIndex)
```

参数说明

❶ anchor：指定图片保存的区域。区域参数的设置在步骤（2）中已经给出了。

❷ pictureIndex：指定图片的索引。图片索引是在步骤（3）中调用 HSSFWorkbook 对象的 addPicture()方法返回的索引值。

设计过程

创建用于操作 Excel 的工具类 ExcelOperationUtil，编写创建 Excel 文档的方法 CreateExcelFile()，然后在方

法中向 Excel 文件中插入图片，关键代码如下：

```
public boolean CreateExcelFile(String filePath,String fileName,String imgPath){
 try{
        HSSFWorkbook workbook = new HSSFWorkbook();          //创建 Excel 工作簿对象
        HSSFSheet sheet = workbook.createSheet();            //在工作簿中创建工作表对象
        workbook.setSheetName(0, "测试");                    //设置工作表的名称
        HSSFRow row1 = sheet.createRow(0);                   //在工作表中创建行对象
        sheet.addMergedRegion(new Region(0,(short)0,0,(short)5)); //合并第 1 行的第 1～第 5 个单元格
        ……//此处省略了其他非关键代码
        FileInputStream fis = new FileInputStream(imgPath);  //创建文件输入流对象
        byte[] imgBytes = new byte[fis.available()];        /创建用于保存图片字节的字节数组
        fis.read(imgBytes);                                  //读取图片字节流，保存到字节数组中
        int picIndex = workbook.addPicture(imgBytes, workbook.PICTURE_TYPE_JPEG); //添加图片字节数据到工作簿对象中
        HSSFClientAnchor anchor = new HSSFClientAnchor();    //创建用于设置图片保存区域的对象
        //设置区域，左上角坐标为第 1 个单元格（5,5），右下角坐标为第 2 个单元格（130,200）
        anchor.setAnchor((short)0, 0, 5, 5, (short)1, 0, 130, 200);
        HSSFPatriarch patri = sheet.createDrawingPatriarch(); //创建用于绘画的对象
        patri.createPicture(anchor , picIndex);              //将图片画到 Excel 工作簿的指定区域
        ……//此处省略了其他非关键代码
        File xlsFile = new File(filePath,fileName);
        FileOutputStream fos = new FileOutputStream(xlsFile);
        workbook.write(fos);                                 //将文档对象写入文件输出流
        fos.close();
        return true;
 } catch (Exception e) {
        e.printStackTrace();
        return false;
 }
}
```

秘笈心法

相对于 JXL 组件而言，POI 组件对插入图片的实现稍微有些麻烦，但是 POI 组件插入图片时，对区域的设置使图片的位置更加灵活。在实际应用中，读者可以根据不同的需求，选择应用哪一个组件来实现插入图片。

实例 562　将数据库数据导出到 Excel 文件

光盘位置：光盘\MR\21\562

高级

实用指数：★★★

实例说明

本实例将介绍如何应用 POI 组件读取数据库中的数据，然后导出 Excel 文件中。运行本实例，如图 21.32 所示，显示了从数据库中读取出的员工表信息，单击“导出”按钮后，这些信息将被导出 Excel 文件中。



图 21.32　将数据库数据导出到 Excel 文件

关键技术

本实例的实现很简单，并没有用到 POI 组件中陌生的方法，用到的方法都已经在前面的实例中介绍过了。

实现的关键之处是：首先将数据库中的数据读取并保存在一个 List 集合中，然后在导出数据的方法中，通过遍历这个 List 集合，再应用 POI 组件的单元格对象 HSSFCell 的 setCellValue()方法，向 Excel 中添加数据即可。

设计过程

（1）创建用于封装员工信息的 JavaBean 类 Employee，关键代码如下：

```
public class Employee {
        private String name;
        private String sex;
        private String dept;
        private String duty;
        private String telephone;
}
```

（2）创建用于操作 Excel 的工具类 ExcelOperationUtil，编写导出数据到 Excel 文件的方法 readDataToExcelFile()，参数 list 中保存了从数据库中读取的封装数据的对象 Employee，关键代码如下：

```
public boolean readDataToExcelFile(List<Employee> list){
 try{
        HSSFWorkbook workbook = new HSSFWorkbook();                        //创建 Excel 工作簿对象
        HSSFSheet sheet = workbook.createSheet();                          //在工作簿中创建工作表对象
        workbook.setSheetName(0, "测试");                                   //设置工作表的名称
        HSSFRow row1 = sheet.createRow(0);                                 //在工作表中创建行对象
        sheet.addMergedRegion(new Region(0,(short)0,0,(short)4));          //合并第 1 行的第 1～第 5 个单元格
        ……//此处省略了一些其他非关键代码
        HSSFCellStyle cellStyle = workbook.createCellStyle();
        cellStyle.setAlignment(CellStyle.ALIGN_CENTER);                    //设置水平居中
        cellStyle.setVerticalAlignment(CellStyle.VERTICAL_CENTER);         //设置垂直居中
        HSSFCell titleCell = row1.createCell(0);
        titleCell.setCellValue("员工信息表");
        ……//此处省略了一些其他非关键代码
        phoneCell.setCellStyle(cellStyle);
        for(int i=0;i<list.size();i++){                                    //遍历保存数据对象的集合
                Employee emp = (Employee)list.get(i);                      //获取封装数据的对象
                HSSFRow dataRow = sheet.createRow(i+2);                    //创建行
                HSSFCell name = dataRow.createCell(0);                     //创建单元格
                name.setCellValue(emp.getName());                          //将数据添加到单元格中
                name.setCellStyle(cellStyle);
                HSSFCell sex = dataRow.createCell(1);
                sex.setCellValue(emp.getSex());
                sex.setCellStyle(cellStyle);
                HSSFCell dept = dataRow.createCell(2);
                dept.setCellValue(emp.getDept());
                dept.setCellStyle(cellStyle);
                HSSFCell duty = dataRow.createCell(3);
                duty.setCellValue(emp.getDuty());
                duty.setCellStyle(cellStyle);
                HSSFCell phone = dataRow.createCell(4);
                phone.setCellValue(emp.getTelephone());
                phone.setCellStyle(cellStyle);
        }
        File xlsFile = new File("E:\\员工信息表.xls");
        FileOutputStream fos = new FileOutputStream(xlsFile);
        workbook.write(fos);                                               //将文档对象写入文件输出流
        fos.close();
        return true;
 } catch (Exception e) {
        e.printStackTrace();
        return false;
 }
}
```

说明： 由于读取数据库的代码比较简单，所以此处省略了这一部分内容。具体代码可以查看本书附赠的光盘，此处不进行详细介绍。

秘笈心法

在实际应用中，根据需求可能需要将数据库中的数据导出 Excel 文件中，此时，可以选择应用 POI 组件来进行处理，应用它可以很方便地将数据导出 Excel 文件中。

实例 563　读取 Excel 文件的数据到数据库

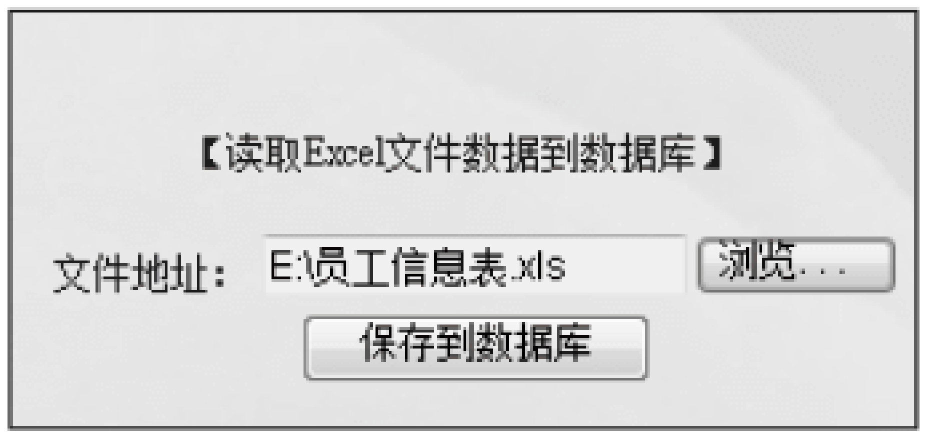
光盘位置：光盘\MR\21\563

高级

实用指数：★★★

实例说明

本实例将介绍如何应用 POI 组件，读取 Excel 文件的数据并保存到数据库。运行本实例，如图 21.33 所示，选择要保存到数据库的 Excel 文件，然后单击“保存到数据库”按钮后，该 Excel 文件的数据将被保存到数据库中。





id	name	sex	dept	duty	telephone
1	张三	男	测试部	经理	13579796464
2	李四	男	研发部	软件工程师	15578781578
3	王二	女	人事部	人事专员	18878963257
4	赵四	女	研发部	项目经理	13388888888

图 21.33　读取 Excel 文件的数据到数据库

关键技术

读取 Excel 文件的数据时，首先需要在创建工作簿对象 HSSFWorkbook 时，在 HSSFWorkbook 类的构造方法中传入一个用于读取文件数据的文件输入流对象，然后应用相应的 getXXX()方法来获取 Excel 文件的数据。

HSSFWorkbook()是用于读取 Excel 文件的构造方法，其语法结构如下：

```
public HSSFWorkbook(POIFSFileSystem fs)
```

参数说明

fs：指定用于管理 Excel 文件的文件对象。在创建读取 Excel 文件的 InputStream 文件输入流时，应用 POIFSFileSystem 对象来对这个文件流进行维护管理。

创建完读取文件的工作簿对象 HSSFWorkbook 之后，应用 getSheetAt()方法即可获取 Excel 文件中指定的工作表对象。getSheetAt()方法的语法结构如下：

```
public HSSFSheet getSheetAt(int index)
```

参数说明

index：指定 Excel 文件中工作表的索引。索引值从 0 开始，也就是第一个工作表。

获取到指定的工作表对象 HSSFSheet 之后，应用其 getXXX()方法即可获取该工作表中的数据。下面介绍这些 getXXX()方法。

- ❑ getFirstRowNum()方法

该方法用于获取指定工作表中数据所在的首行行号。行号值从 0 开始，也就是说，如果工作表中的首行数据在第 3 行，那么该方法返回值为 2。

- ❑ getLastRowNum()方法

该方法用于获取指定工作表中数据所在的末行行号。行号值从 0 开始，也就是说，如果工作表中的末行数据在第 11 行，那么该方法返回值为 10。

- ❑ getRow()方法

该方法用于获取工作表中的指定行，其语法结构如下：

```
public HSSFRow getRow(int rowIndex)
```

参数说明

rowIndex：指定行所在的索引。索引值从 0 开始。

说明： HSSFSheet 类中还包含一些其他的 getXXX()方法，在此不再详细介绍。具体方法说明请参见 POI 组件的 API 文档。

设计过程

（1）创建用于封装员工信息的 JavaBean 类 Employee，关键代码如下：

```
public class Employee {
	private String name;
	private String sex;
	private String dept;
	private String duty;
	private String telephone;
}
```

（2）创建用于操作 Excel 的工具类 ExcelOperationUtil，编写读取 Excel 文件数据的方法 readExcelFileToDB()，参数 filePath 指的是 Excel 文件的路径。在该方法中，读取 Excel 文件的数据，将每一行数据封装在 Employee 对象中，然后再循环每一行，将封装好的每一行的 Employee 对象添加到 List 集合中，关键代码如下：

```
public List<Employee> readExcelFileToDB(String filePath){
 List<Employee> list = new ArrayList<Employee>();
 try{
	FileInputStream fis = new FileInputStream(filePath);
	POIFSFileSystem fs   = new POIFSFileSystem(fis);
	HSSFWorkbook workbook = new HSSFWorkbook(fs);		//创建 Excel 工作簿对象
	HSSFSheet sheet = workbook.getSheetAt(0);		//获取第 1 个工作表
	for(int i=2;i<=sheet.getLastRowNum();i++){		//循环 Excel 文件的每一行
		Employee emp = new Employee();
		HSSFRow row = sheet.getRow(i);			//获取第 i 行
		HSSFCell cell1 = row.getCell(0);
		HSSFCell cell2 = row.getCell(1);
		HSSFCell cell3 = row.getCell(2);
		HSSFCell cell4 = row.getCell(3);
		HSSFCell cell5 = row.getCell(4);
		String name = cell1.getStringCellValue();		//获取第 i 行的第 1 个单元格的数据
		emp.setName(name);
		String sex = cell2.getStringCellValue();		//获取第 i 行的第 2 个单元格的数据
		emp.setSex(sex);
		String dept = cell3.getStringCellValue();		//获取第 i 行的第 3 个单元格的数据
		emp.setDept(dept);
		String duty = cell4.getStringCellValue();		//获取第 i 行的第 4 个单元格的数据
		emp.setDuty(duty);
		String phone = cell5.getStringCellValue();		//获取第 i 行的第 5 个单元格的数据
		emp.setTelephone(phone);
		list.add(emp);
	}
	fis.close();
	return list;
 } catch (Exception e) {
	e.printStackTrace();
	return null;
 }
}
```

说明： 由于将 List 集合的数据添加到数据库的方法比较简单，所以此处省略了这一部分内容。具体代码可以查看本书附赠的光盘，此处不再进行详细介绍。

秘笈心法

在实际应用中，一个 Excel 文件中可能包含多行数据，这样在读取时就应该应用循环来实现，而关键之处就是确定循环范围。应用工作表对象 HSSFSheet 的 getLastRowNum()方法可以获取到最后一行的行号，因此在循环中使用该方法就可以限定循环范围。

实例 564　设置 Excel 文件的打印属性

光盘位置：光盘\MR\21\564

高级　实用指数：★★★

实例说明

本实例将介绍如何应用 POI 组件，对 Excel 工作表的打印属性进行设置。运行本实例，将数据导出 Excel 文件之后，设置打印属性。如图 21.34 所示为在打印预览中的显示效果。



员工姓名	员工性别	所在部门	职务	联系电话
张三	男	测试部	经理	13579796464
李四	男	研发部	软件工程师	15578781578
王二	女	人事部	人事专员	18878963257
赵四	女	研发部	项目经理	13388888888
员工1	男	测试部	测试人员	13579796464
员工2	男	研发部	软件工程师	15578781578
员工3	女	人事部	人事专员	18878963257
员工4	女	研发部	程序员	13388888888

图 21.34　设置 Excel 工作表打印属性之后的打印预览效果

关键技术

应用 POI 组件设置 Excel 工作表的打印属性时，需要用到 HSSFWorkbook、HSSFSheet 和 HSSFPrintSetup 这 3 个类中的方法进行设置。下面对这几个类中关于设置打印属性的方法进行介绍。

1. HSSFWorkbook 类

该类中包含一个用于设置打印区域的 setPrintArea()方法，其语法结构如下：

```
public void setPrintArea(int sheetIndex,int startColumn,int endColumn,int startRow,int endRow)
```

参数说明

❶ sheetIndex：指定要设置打印区域的工作表索引。索引值从 0 开始。

❷ startColumn：指定打印区域的起始列索引。

❸ endColumn：指定打印区域的结束列索引。

❹ startRow：指定打印区域的起始行索引。

❺ endRow：指定打印区域的结束行索引。

setPrintArea()方法还包含一个重载方法，其语法结构如下：

```
public void setPrintArea(int sheetIndex,java.lang.String reference)
```

参数说明

❶ sheetIndex：指定要设置打印区域的工作表索引。索引值从 0 开始。

❷ reference：指定打印区域的规则字符串。该规则字符串的写法格式如$A$1:$B$2。

2. HSSFSheet 类

该类中包含了一些用于设置打印属性的方法，这些方法的说明如表 21.3 所示。

表 21.3　HSSFSheet 类中常用的设置打印属性的方法

方　　法	说　　明
setPrintGridlines(boolean newPrintGridlines)	设置是否打印网格线
setHorizontallyCenter(boolean value)	设置是否水平居中
setVerticallyCenter(boolean value)	设置是否垂直居中

续表

方　　法	说　　明
getHeader()	返回用于设置页眉的 HSSFHeader 对象。该对象中包含的 setLeft()、setCenter()和 setRight()方法，用于设置左侧、中间和右侧的页眉内容
getFooter()	返回用于设置页脚的 HSSFFooter 对象。该对象中包含的 setLeft()、setCenter()和 setRight()方法，用于设置左侧、中间和右侧的页脚内容

3. HSSFPrintSetup 类

在设置打印属性时，主要应用这个类的方法，其常用方法的说明如表 21.4 所示。

表 21.4　HSSFPrintSetup 类的常用方法

方　　法	说　　明
setFitHeight(short height)	设置页高
setFitWidth(short width)	设置页宽
setFooterMargin(double footermargin)	设置页脚边距
setHeaderMargin(double headermargin)	设置页眉边距
setLandscape(boolean ls)	设置打印方向为横向或纵向。取值为 false 为纵向，true 为横向
setNoColor(boolean mono)	设置打印时是否带有颜色
setPaperSize(short size)	设置打印纸大小。取值为 HSSFPrintSetup 类的静态常量

设计过程

（1）创建用于操作 Excel 的工具类 ExcelOperationUtil，编写设置打印属性的 setPrint()方法，参数 sheet 指的是要设置打印属性的工作表对象，关键代码如下：

```
public HSSFSheet setPrint(HSSFSheet sheet){
        HSSFHeader header = sheet.getHeader();
        header.setRight("页眉");                                    //添加右侧页眉
        HSSFHeader.fontSize((short)8);                              //设置字号
        HSSFFooter footer = sheet.getFooter();
        HSSFFooter.fontSize((short)8);                              //设置字号
        footer.setRight("页脚");                                    //添加右侧页脚
        sheet.setPrintGridlines(true);                              //打印网格线
        HSSFPrintSetup printSet = sheet.getPrintSetup();
        printSet.setFitWidth((short)2);                             //设置页宽
        printSet.setFitHeight((short)2);                            //设置页高
        printSet.setPaperSize(HSSFPrintSetup.A4_PAPERSIZE);         //设置打印纸大小
        printSet.setHeaderMargin(5.5);                              //设置页眉边距
        printSet.setFooterMargin(5.5);                              //设置页脚边距
        sheet.setVerticallyCenter(true);                            //设置垂直居中
        sheet.setHorizontallyCenter(true);                          //设置水平居中
        return sheet;
}
```

（2）在导出数据的 readDataToExcelFile()方法中，调用 setPrint()方法设置打印属性，关键代码如下：

```
public boolean readDataToExcelFile(List<Employee> list){
        try{
                HSSFWorkbook workbook = new HSSFWorkbook();         //创建 Excel 工作簿对象
                HSSFSheet sheet = workbook.createSheet();           //在工作簿中创建工作表对象
                sheet = this.setPrint(sheet);                       //调用设置打印属性的方法，设置工作表的打印属性
                workbook.setSheetName(0, "测试");                   //设置工作表的名称
                HSSFRow row1 = sheet.createRow(0);                  //在工作表中创建行对象
                sheet.addMergedRegion(new Region(0,(short)0,0,(short)4)); //合并第 1 行的第 1～5 个单元格
                HSSFCell titleCell = row1.createCell(0);
                titleCell.setCellValue("员工信息表");
                ……//此处省略了其他非关键代码
```

```
            File xlsFile = new File("E:\\员工信息表.xls");
            FileOutputStream fos = new FileOutputStream(xlsFile);
            workbook.write(fos);                                    //将文档对象写入文件输出流
            fos.close();
            return true;
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }
```

■秘笈心法

在实际应用中，有时需要在程序中直接完成对 Excel 工作表的打印，那么在打印之前就需要首先控制工作表的打印属性设置，然后再完成打印。

第22章

报表与打印

- Web 打印
- 利用 Word 打印报表
- 利用 Excel 打印报表
- 应用 WebBrowser+CSS 套打邮寄产品单
- 打印库存报表
- 高级报表

22.1　Web 打印

Web 打印是一种常用的打印方式，其使用方法简单、方便、快捷，用户在浏览网页的同时就可以实现打印的功能。下面介绍几个利用 Web 打印的实例。

实例 565　利用 JavaScript 调用 IE 自身的打印功能

初级

光盘位置：光盘\MR\22\565

实用指数：★★★

实例说明

利用 JavaScript 调用 IE 自身的打印功能实现打印，这种方法比较简单，也是常用的打印方式。使用该方法只需将要打印的页面设计好，再通过 JavaScript 的 window 对象的 print()方法调用 IE 的打印功能即可。运行本实例，单击“打印”超链接后会弹出“打印”对话框，如图 22.1 所示，然后进行相应的设置，并进行打印。



图 22.1　利用 JavaScript 调用 IE 自身的打印功能实现打印

关键技术

本实例主要通过调用 window 对象的打印方法 print()来实现打印功能。window 对象的 print()方法的语法格式如下：

```
window.print();
```

例如：

```
<a href="#" onClick="window.print()">打印</a>
```

设计过程

（1）从数据表中获取要打印的数据并以列表的形式显示。

（2）调用 window 对象的打印方法实现打印功能，其关键代码如下：

```
<a href="#" onClick="window.print()">打印</a>
```

秘笈心法

调用 JavaScript 的 window 对象的 print()方法非常简单，能快速地实现网页数据的打印，但是它不能像 Excel 以及 Word 中的打印功能那样可以方便地调整打印的格式。因此，对打印格式要求较高的情况，可以考虑不用这种打印的实现方式。

实例 566　利用 WebBrowser 打印

光盘位置：光盘\MR\22\566

初级　实用指数：★★★

实例说明

WebBrowser 是 IE 内置的浏览器控件，无须用户下载。它的优点是客户端独立完成打印目标文档的生成，减轻服务器负荷；缺点是源文档的分析操作复杂，并且要对源文档中要打印的内容进行约束。运行本实例，单击“打印预览”超链接，即可打开“打印预览”对话框，如图 22.2 所示。单击“打印”超链接即可打开“打印”对话框进行打印。



图 22.2　利用 WebBrowser 打印

关键技术

本实例主要应用 IE 内置的 WebBrowser 控件实现，该控件的具体参数如下。

- document.all.WebBrowser.Execwb(7,1)：表示打印预览。
- document.all.WebBrowser.Execwb(6,1)：表示打印。
- document.all.WebBrowser.Execwb(6,6)：表示直接打印。
- document.all.WebBrowser.Execwb(8,1)：表示页面设置。

设计过程

（1）在页面中添加<object>标签，调用 WebBrowser 控件，其关键代码如下：

```
<object id="WebBrowser" classid="ClSID:8856F961-340A-11D0-A96B-00C04Fd705A2" width="0" height="0">
</object>
```

（2）建立相关的“打印”超链接，并调用 WebBrowser 控件的相应参数实现打印预览、打印等功能，其关键代码如下：

```
<a href="#" onClick="document.all.WebBrowser.Execwb(7,1)">打印预览</a>
<a href="#" onClick="document.all.WebBrowser.Execwb(6,1)">打印</a>
<a href="#" onClick="document.all.WebBrowser.Execwb(6,6)">直接打印</a>
<a href="#" onClick="document.all.WebBrowser.Execwb(8,1)">页面设置</a>
```

秘笈心法

在 HTML 中应用<object>标签可以将外部 ActiveX 控件加载到网页中；应用<object>标签可以调用系统中的 ActiveX 控件应用程序。其中 classid 属性值是生成 ActiveX 控件时的唯一注册编号。在网页中加入系统中的控件时，classid 是必须要添加的，根据此 id 才可以找到相应的控件应用程序。

实例 567　打印分组报表

光盘位置：光盘\MR\22\567

初级　实用指数：★★★

实例说明

分组报表是根据表中的某个条件在另一张表中查询与此内容相关的一系列信息，并将此类信息打印成报表。

分组报表在实际中应用广泛。在开发药品后台管理系统时，通常要求以药品种类分组显示药品销售情况，在这种情况下，就可以应用分组报表打印。本实例将以 WebBrowser 打印技术演示分组报表，通过药品相关信息表中的药品编号与药品销售表中的药品号相关联查询数据，以达到分组的目的。实例运行结果如图 22.3 所示。



图 22.3　药品销售管理系统分组报表打印

关键技术

在 JSP 页面中可以调用 IE 内置的浏览器控件 WebBrowser，通过使用 WebBrowser 控件的 Execwb 方法可实现打印预览、打印、页面设置等操作。

设计过程

（1）首先，为了取得数据表中的值，创建 JavaBean，名称为 Tdbb.java。实现 Tdbb.java 除了需要使用基本的 setXXX()和 getXXX()方法之外，还需使用一个根据条件取数据的 getReport()方法，此方法的参数是用户输入的产地，getReport()方法的返回值类型为集合 Collection，方便 JSP 取值时使用，程序代码如下：

```
public Collection getRecord(){
        Collection ret=new ArrayList();                          //实例化集合
        String sql="select * from tb_yptable";                   //查找数据
        ConnDB conn=new ConnDB();                                //创建数据库连接
        ResultSet rs=conn.executeQuery(sql);                     //执行查询
        try{
                while(rs.next()){                                //遍历结果集
                        Tdbb t=new Tdbb();                       //实例化实体对象
                        t.setId(rs.getString(1));
                        ……                                       //为对象赋值属性
                        ret.add(t);                              //将对象添加到集合中
        }
        conn.close();                                            //关闭数据库连接
        }catch(Exception e){
                e.printStackTrace();
        }
        return ret;                                              //返回集合
}
```

（2）取出药品相关信息表中的内容放入 Collection 中，以便于显示，程序代码如下：

```
Collection ret=(Collection)t.getRecord();
```

（3）根据药品编号条件将两个表联系在一起，在 SQL 语句中用左外连接把两个表联系起来，根据报表编号取得结果集，此方法与 getRecord()方法采用了重载技术，具体程序代码如下：

```
public Collection getRecord(String ypbh){
        Collection ret=new ArrayList();                          //实例化集合
        //查询语句
        String sql="SELECT t1.ypph, t1.number, t1.dj, t1.je, t1.khqc ,t1.ypbh FROM tb_xsbb t1 LEFT OUTER
                JOIN tb_yptable t2 ON t1.ypbh = t2.ypid where ypbh='"+ypbh+"'";
        ConnDB conn=new ConnDB();                                //创建数据库连接
        ResultSet rs=conn.executeQuery(sql);                     //执行查询
        try{
                while(rs.next()){                                //遍历查询的结果集
                        Tdbb t=new Tdbb();                       //实例化实体对象
                        t.setJe(rs.getString("je"));
                        ……                                       //为实体对象的属性赋值
                        ret.add(t);                              //将实体对象添加到集合当中
                }
                conn.close();                                    //关闭数据库连接
        }catch(Exception e){
                e.printStackTrace();
        }
        return ret;                                              //返回集合
}
```

（4）在 JSP 中设置 JavaScript 打印按钮，程序代码如下：

```
<table align="center" id="pay">
<tr class="print">
<td >
        <INPUT type="button" value="打印设置" id="button1" onclick="setPrint();">
        <INPUT type="button" value="打印预览" id="button2" onclick="previewPrint();">
        <INPUT type="button" value="打印"    id="button3"    onclick="window.print();">
</td>
</tr>
</table>
```

（5）注意<head></head>之间要放入 JavaScript 代码，程序代码如下：

```
<script language="Javascript">
function     previewPrint(){
        WB.ExecWB(7,1)
}
function     setPrint(){
        WB.ExecWB(8,1);
}
</script>
```

（6）注意需在<body>之间放入以下代码，上述 JavaScript 才可以使用。

```
<OBJECT classid="CLSID:8856F961-340A-11D0-A96B-00C04FD705A2" height="0" id="WB" width="0"> </OBJECT>
```

秘笈心法

在 JSP 中使用迭代函数列出数据较其他方案取值有一些特别，首先取出药品相关信息表的相关数据，并且打印在 JSP 界面上，然后根据药品票号取出药品销售信息表的相关数据，也就是使用了嵌套循环的方式实现统计报表功能。

22.2　利用 Word 打印报表

Microsoft Office 办公软件中的 Word 是 Microsoft 提供的文档处理软件，它在处理、打印文档及资料过程中所显现出来的强大功能是有目共睹的。本节将介绍如何将网页中的数据直接导出 Word 中，进行直接打印或处理后再进行打印。

实例 568	将页面中的客户列表导出到 Word 并打印 光盘位置：光盘\MR\22\568	初级 实用指数：★★★

实例说明

在开发 Web 应用程序时，经常会遇到打印页面中的部分内容的情况，这时可以将这部分内容导出 Word，然后再打印。本实例将介绍如何将页面中的客户列表导出到 Word 并打印。运行本实例，在页面中将显示客户信息列表，单击“打印”超链接后，将把 Web 页中的数据导出 Word 的新建文档中，如图 22.4 所示，并保存在 Word 的默认文档保存路径中，最后调用打印程序打印该文档。

关键技术

本实例主要应用 JavaScript 的 ActiveXObject()构造函数创建一个 OLE Automation ActiveX 对象的实例，并应用该实例的相关方法实现。

ActiveXObject()构造函数的一般语法格式如下：

```
var objectVar = new ActiveXObject(class[, servername]);
```

参数说明

❶ objectVar：用于指定引用对象的变量。

❷ class：用于指定应用程序的名字或包含对象的库，并且指定要创建的对象类的类型。采用 library.object 的语法格式，如 Word.Application，表明要创建的是 Word 对象。

❸ servername：可选参数，用于指定包含对象的网络服务器的名字。

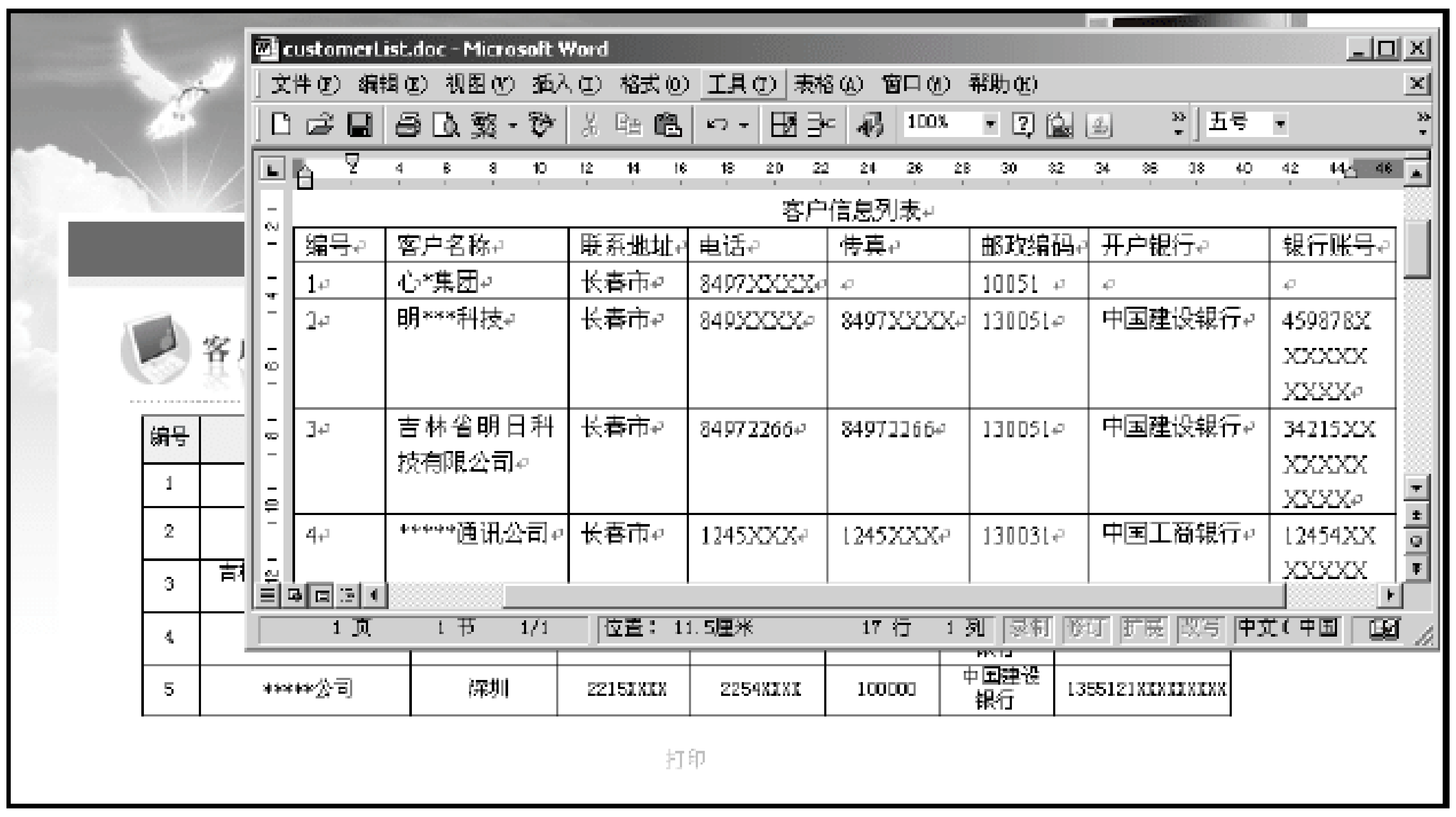


图 22.4　将页面中的客户列表导出到 Word 并打印

设计过程

（1）将显示客户信息的表格的 id 设置为 customer，因为要打印此表中的数据，其关键代码如下：

```
<table id="customer" width="650" border="0" align="center" cellspacing="1" bgcolor="#000000">
```

（2）编写自定义 JavaScript 函数 outDoc()，用于将 Web 页面中的客户列表信息导出 Word，并进行自动打印，其关键代码如下：

```
function outDoc(){
    var table=document.all.customer;                    //获取表格对象
    row=table.rows.length;                              //获取表格的行数
    column=table.rows(1).cells.length;                  //获取表格的列数
    var wdapp=new ActiveXObject("Word.Application");    //创建 Word Application 对象
    wdapp.visible=true;                                 //设置 Word 应用程序显示
    wddoc=wdapp.Documents.Add();                        //添加新的文档
    thearray=new Array();
    //将页面中表格的内容存放在数组中
    for(i=0;i<row;i++){
        thearray[i]=new Array();
        for(j=0;j<column;j++){
            thearray[i][j]=table.rows(i).cells(j).innerHTML;
        }
    }
    var range = wddoc.Range(0,0);                       //创建第一行对象
    range.Text="客户信息列表"+"\n";                      //设置此行的文字
    range.ParagraphFormat.Alignment=1;                  //设置行文字样式，居中显示，默认为居左，如果值为 2，则表示居右
    wddoc.Range(1,1);                                   //创建第二行对象
    wdapp.Application.Activedocument.Paragraphs.Add(range);   //添加一个段落，内容为行 Range 对象中的内容
    wdapp.Application.Activedocument.Paragraphs.Add();        //添加一个空的段落
    rngcurrent=wdapp.Application.Activedocument.Paragraphs(3).Range; //获取第 3 段的 Range 对象
    var objTable=wddoc.Tables.Add(rngcurrent,row,column)      //插入表格
    for(i=0;i<row;i++){
        for(j=0;j<column;j++){                          //循环将数组的值添加到 Word 文档的表格中
            objTable.Cell(i+1,j+1).Range.Text = thearray[i][j].replace(" ","");
        }
    }
    //保存此 Word 文档内容
    wdapp.Application.ActiveDocument.SaveAs("customerList.doc",0,false,"",true,"",false,false,false,false,false);
    wdapp.Application.Printout();                       //调用 Word 自动打印功能
    wdapp=null;
}
```

（3）通过单击“打印”超链接调用自定义 JavaScript 函数 outDoc()，其关键代码如下：

```
<a href="#" onClick="outDoc();">打印</a>
```

秘笈心法

当 new ActiveXObject()对象的参数设置为 Word.Application 时，此对象即表示一个 Word 文档对象，然后调

用该对象的相应方法或属性就可以操作 Word 文档。

实例 569　利用 Word 自动打印指定格式的会议记录　初级

光盘位置：光盘\MR\22\569　实用指数：★★★

实例说明

在开发网络应用程序时，有时需要对输入的信息按指定的格式进行打印。例如，在办公自动化系统中，录入的会议记录信息就需要按指定的格式打印。运行本实例，在页面中输入相应的会议信息，单击“Word 打印”按钮，即可将录入的会议信息导出到指定的 Word 文档中，并自动按该文档指定的格式打印。实例运行结果如图 22.5 所示。

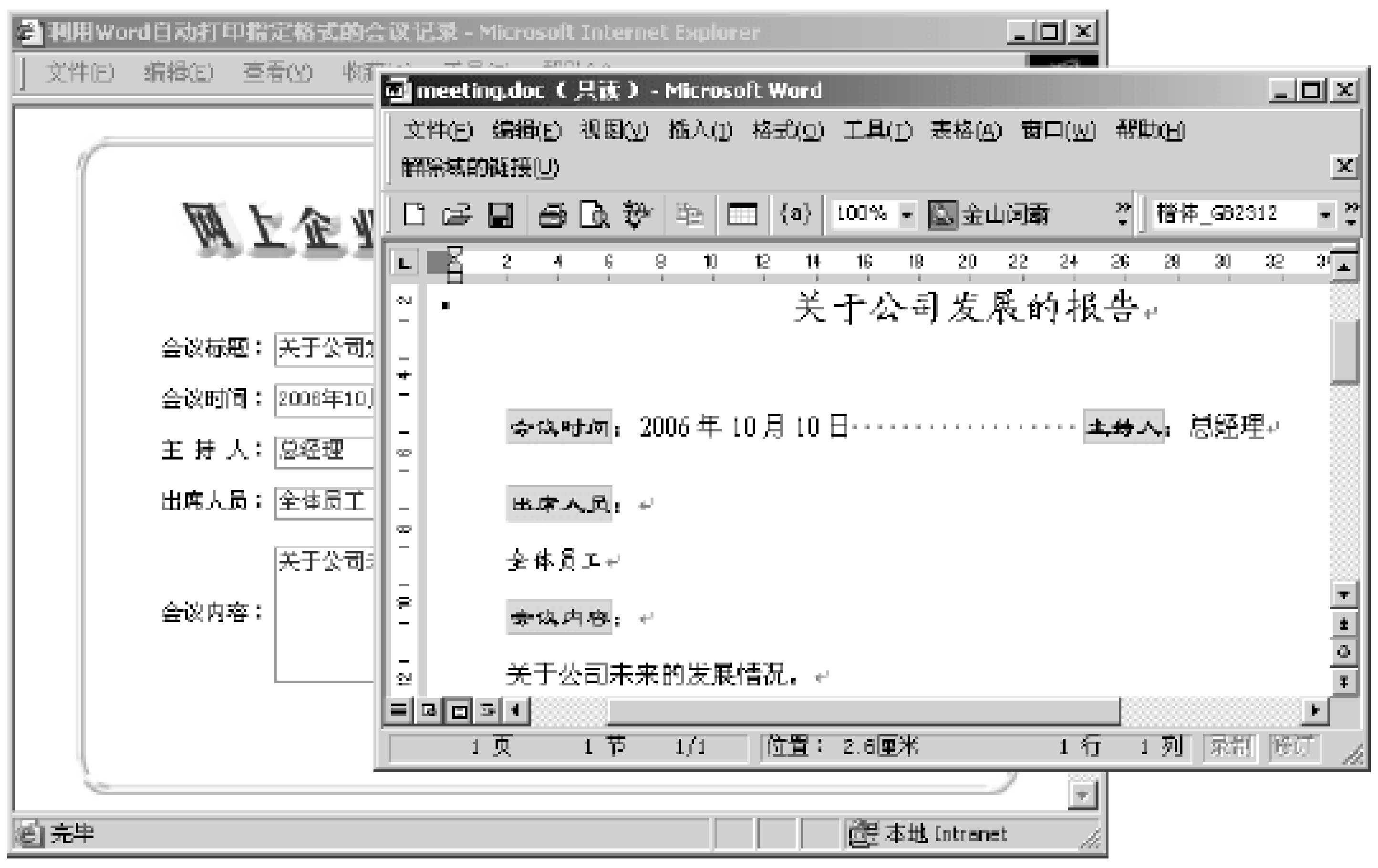


图 22.5　利用 Word 自动打印指定格式的会议记录

关键技术

实现利用 Word 自动打印指定格式的会议记录的思路如下：

（1）应用 JavaScript 的 ActiveXObject()构造函数创建一个 Word Application 对象的实例。

（2）打开指定的 Word 文档。

（3）通过 Word Application 对象的表示书签的 Bookmarks 集合的相应方法将表单内容写入到指定的 Word 文档中。

（4）调用 wdapp.Application.Printout()方法实现自动打印 Word 文档的功能。

设计过程

（1）创建一个 Word 文档，在该文档中设计好要打印的会议记录的格式，并将其保存到实例根目录下，名称为 meeting.doc。

（2）在创建好的 Word 文档中的指定位置插入书签。插入书签的方法如下：首先选中需要替换的文本，然后选择“插入”→“书签”命令，在打开的对话框中输入书签名，并单击“添加”按钮即可。

（3）在实例主页面中添加用于收集会议信息的表单及表单元素，关键代码如下：

```
<form name="form1" method="post" action="">
  <table width="551" height="380" border="0" align="center" cellspacing="0" >
    <tr align="center"><td height="114"   colspan="2"> </td></tr>
    <tr >
      <td width="127" height="27" align="right">会议标题：</td>
      <td width="420" align="left" ><input name="title" type="text" id="title" value="关于公司发展的报告" size="50"></td>
    </tr>
    <tr >
      <td width="127" height="27" align="right">会议时间：</td>
      <td align="left" ><input name="meetingTime" type="text" id="meetingTime" value="2009 年 03 月 17 日"></td>
    </tr>
```

```
    <tr >
      <td height="27" align="right" >主  持  人：</td>
      <td align="left" ><input name="compere" type="text" id="compere" value="总经理"></td>
    </tr>
    <tr >
      <td height="27" align="right" >出席人员：</td>
      <td align="left" ><input name="attend" type="text" id="attend" value="全体员工" size="58"></td>
    </tr>
    <tr >
      <td height="90" align="right" >会议内容：</td>
      <td align="left" ><textarea name="content" cols="56" rows="5" class="wenbenkuang" id="content">关于公司未来的发展情况。</textarea>
</td>
    </tr>
    <tr align="center">
      <td height="53" colspan="2"><input name="Submit" type="button" class="btn_grey" onClick="outDoc()" value="Word 打印"></td>
    </tr>
  </table>
</form>
```

（4）编写自定义 JavaScript 函数 outDoc()，用于将表单收集的数据导出 Word，并进行自动打印，其关键代码如下：

```
function outDoc(){
        var wdapp=new ActiveXObject("Word.Application");          //创建 Word 文档对象
        wdapp.visible=true;                                        //显示此 Word 文档对象
        wddoc=wdapp.Documents.Open("<%=request.getRequestURL()%>meeting.doc");    //打开指定的 Word 文档
        var form=document.all.form1;                               //获取页面表单对象
        title=form.title.value;                                    //获取表单中的会议标题
        meetingTime=form.meetingTime.value;                        //获取表单中的会议时间
        compere=form.compere.value;                                //获取表单中的主持人
        attend=form.attend.value;                                  //获取表单中的出席人员
        content=form.content.value;                                //获取表单中的会议内容
        range =wdapp.ActiveDocument.Bookmarks("title").Range;
        range.Text=title;                                          //将标题输出到 Word
        range =wdapp.ActiveDocument.Bookmarks("meetingTime").Range;
        range.Text=meetingTime;                                    //将会议时间输出到 Word
        range =wdapp.ActiveDocument.Bookmarks("compere").Range;
        range.Text=compere;                                        //将会议主持人输出到 Word
        range =wdapp.ActiveDocument.Bookmarks("attend").Range;
        range.Text=attend;                                         //将出席人员输出到 Word
        range =wdapp.ActiveDocument.Bookmarks("content").Range;
        range.Text=content;                                        //将会议内容输出到 Word
        wddoc.Application.Printout();                              //调用 Word 自动打印方法
        wdapp=null;
}
```

（5）通过单击“Word 打印”按钮调用自定义 JavaScript 函数 outDoc()，其关键代码如下：

```
<input name="Submit" type="button" class="btn_grey" onClick="outDoc()" value="Word 打印">
```

秘笈心法

在浏览器中应用 JavaScript 的 ActiveXObject()构造函数调用 Word 文档程序时，需要对浏览器的安全级别进行设置，否则默认情况下不允许访问。打开浏览器，选择“工具”→“Internet 选项”命令，弹出“Internet 选项”对话框，选择“安全”选项卡，单击“自定义级别”按钮，在弹出的对话框中选中“对为标记为可安全执行脚本的 ActiveX 控件初始化并执行脚本”复选框，最后重新启动浏览器再执行本实例即可调用 Word 程序。

实例 570	利用 Word 生成的 HTML 实现打印 光盘位置：光盘\MR\22\570	初级 实用指数：★★★

实例说明

由于 Word 支持 HTML 格式，所以可以先在 Word 中做好模板，另存为 Web 页面，然后将 HTML 文件修改

为 JSP 文件进行操作。在开发游戏网后台管理系统时，通常需要保留浏览者注册的相关信息，如图 22.6 所示。为了方便后台管理者的操作，需要把数据导出到 Word 中进行打印操作，在这种情况下，就可以使用 Word 打印报表，如图 22.7 所示。

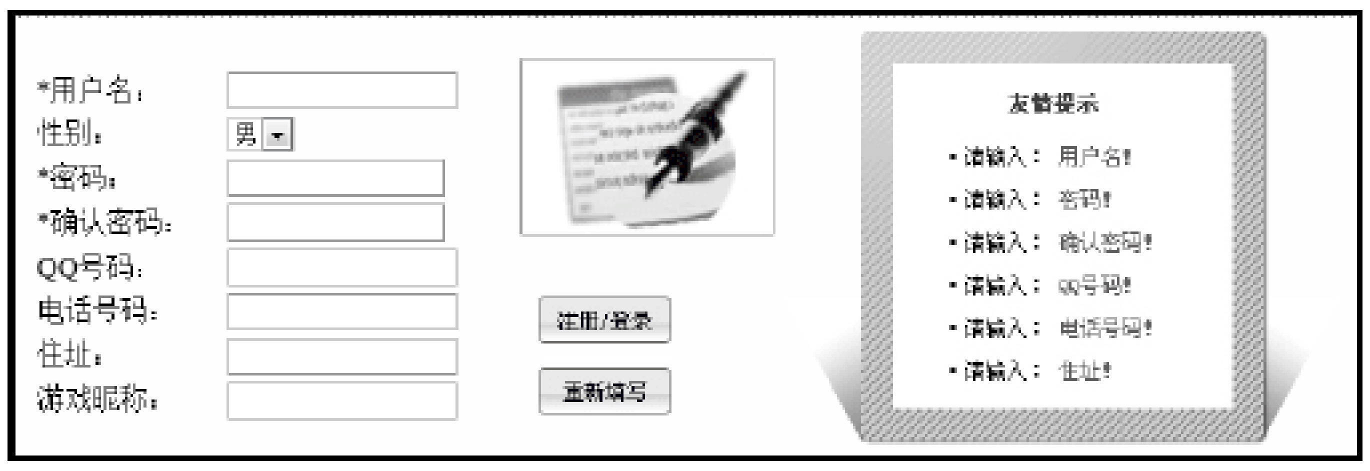


图 22.6　游戏网用户注册页面

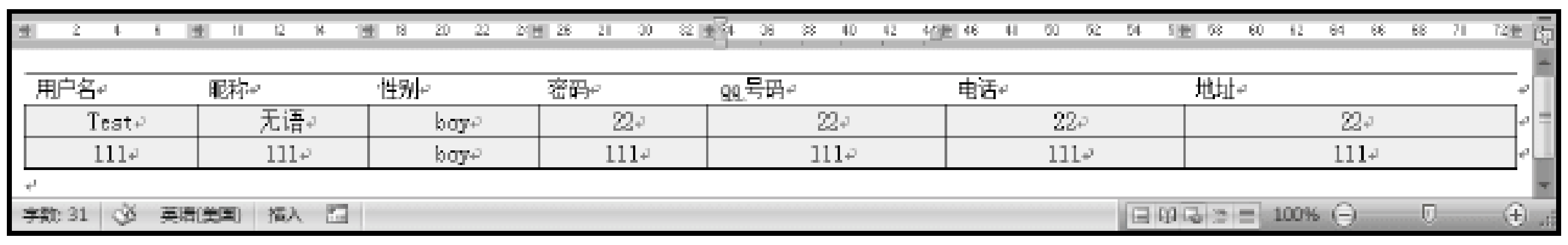


图 22.7　打印 Word 报表

关键技术

本实例实现的思路是：首先需要在 Word 中做好表格模板并另存为 Web 格式，然后将生成的 HTML 文件修改为 JSP 文件，接下来通过自定义标签读取数据库中的数据并显示在此 JSP 中，最后运行此 JSP 就会以 Word 形式打开。在 JSP 中，需要应用 response 内置对象的 setContentType()方法设置响应类型为 application/msword，其含义是响应正文为 Word，这样即可通过 JSP 生成 Word。

设计过程

（1）首先从数据库中取出数据，利用 JavaBean 的 setXXX()方法把数据存入 List 中，程序代码如下：

```
public List ReadRecord(){
        Conn conn=new Conn();                                   //创建数据库连接
        String sql="select * from wordtest order by id";        //查找注册信息
        ResultSet rs=conn.executeQuery(sql);                    //执行查询
        List l=new ArrayList();                                 //实例化 List 集合
        try{
                while(rs.next()){                               //遍历结果集
                        WordTest t=new WordTest();
                        ……                                      //把数据存入 JavaBean 中
                        l.add(t);                               //将信息放入 List 集合中
                }
        }catch(Exception e){
                e.printStackTrace();
        }
        return l;                                               //返回集合
        }
}
```

（2）在 Word 中添加表格信息，另存为 Web 格式，文件名为 test.html，然后修改为 test.jsp。当然，客户端自身也要有 Word 软件才可以打开。保存后的 JSP 部分程序代码如下：

```
<%@ page contentType="application/msword;charset=GBK" %>
<%@ page import="Java.sql.*" %>
<%@taglib uri="/WEB-INF/mytag.tld" prefix="mytag" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html xmlns:o="urn:schemas-microsoft-com:office:office"
xmlns:w="urn:schemas-microsoft-com:office:word"
xmlns="http://www.w3.org/TR/REC-html40">
<head>
<meta http-equiv=Content-Type content="text/html; charset=GB2312">
<meta name=ProgId content=Word.Document>
<meta name=Generator content="Microsoft Word 9">
<meta name=Originator content="Microsoft Word 9">
<link rel=File-List href="./test.files/filelist.xml">
<title>用户名</title>
<!--[if gte mso 9]><xml>
```

（3）为了使 JSP 界面没有逻辑代码，在这里使用自定义标签技术，创建一个 displayTag.java 类作为自定义标签类来显示内容。displayTag.java 类继承了 TagSupport 类，重写了 doEngTag()方法，可以在 displayTag.java 中实例化 out 和 request 等内置对象，这样，自定义类就可以像 Servlet 一样把一些内容显示在 JSP 上，调用 JavaBean

的 ReadRecord()方法，返回值为 List 类型，根据 JavaBean 的 getXXX()方法打印数据库数据表中取出的值，部分程序代码如下：

```
public class displayTag extends TagSupport{
public int doEndTag() throws JSPException{
 JSPWriter out=pageContext.getOut();
 HttpServletRequest request=(HttpServletRequest)pageContext.getRequest();
        WordTest w=new WordTest();                      //实例化对象
        List l=w.ReadRecord();                          //获取信息
        try{
                if(l.size()>0&&!l.isEmpty()){           //如果 List 集合不为空
                for(int i=0;i<l.size();i++){            //遍历 List 集合
                        WordTest w2=(WordTest)l.get(i); //取出对象
                        out.println("<tr>");
                        out.println("<td width=189 valign=top style='width:142.0pt;border:none;
                        border-bottom:solid windowtext .5pt;padding:0cm 5.4pt 0cm 5.4pt'>");
                        ……                              //写出循环打印表格
                return super.doEndTag();
        }
}
}
```

（4）若想在 JSP 界面引用自定义标签，首先需要在 WEB-INF 目录下创建 mytag.tld 标签文件，<name></name>之间代表标签的名字，<tagclass></tagclass>之间代表自定义标签类所在位置，关键代码如下：

```
<tag>
        <name>display</name>#自定义标签名字
        <tagclass>com.wsy.displayTag</tagclass>#自定义标签所在位置
        <bodycontent>empty</bodycontent>
        <info>A demo</info>
</tag>
```

（5）在 test.jsp 文件中加入如下程序代码，就可以在 test.jsp 中引用自定义标签。

```
<%@taglib uri="/WEB-INF/mytag.tld" prefix="mytag" %> <!-- 在 jsp 中声明自定义标签 -->
```

（6）在 test.jsp 文件中需要显示表格的位置加入自定义标签，就可以正常显示表格，程序代码如下：

```
<mytag:display/><!-- 在 jsp 页面中引用自定义标签 -->
```

秘笈心法

在 JSP 页面中使用自定义标签可以简化 JSP 页面代码量，使逻辑代码和前台代码充分分离，有助于页面的后期维护，自定义标签继承了 TagSupport 类，重写了父类方法 doEndTag()，可以实例化一个 out，out.write()将数据输出到页面中。同时，需要在 WEB-INF 中建立*.tld 文件，这样可在 JSP 中引用标签文件。

22.3 利用 Excel 打印报表

Microsoft Excel 是 Microsoft 提供的用于办公管理的应用软件，其强大的表格统计功能是其他同类型软件无法比拟的。本节将介绍如何将 Web 页中的数据信息导出到 Excel 中并打印，以提高用户的办公效率。

实例 571	利用 Excel 打印工作报表 光盘位置：光盘\MR\22\571	初级 实用指数：★★★

实例说明

为了方便用户操作，有时需要将页面中的数据导出到 Excel 中进行处理，并利用其强大的打印功能实现页面数据的打印。下面将介绍如何利用 Excel 打印工资报表。运行本实例，首先在“请选择要打印的工资月份”下拉列表框中选择 2006-09，单击“查询”按钮后，在下面的表格中将显示符合条件的工资信息，然后单击“打印”超链接，可以将查询结果列表导出到 Excel 中（用户可以对导出的数据进行处理），最后通过 Excel 自身的

打印功能实现工资报表的打印。程序运行结果如图 22.8 和图 22.9 所示。

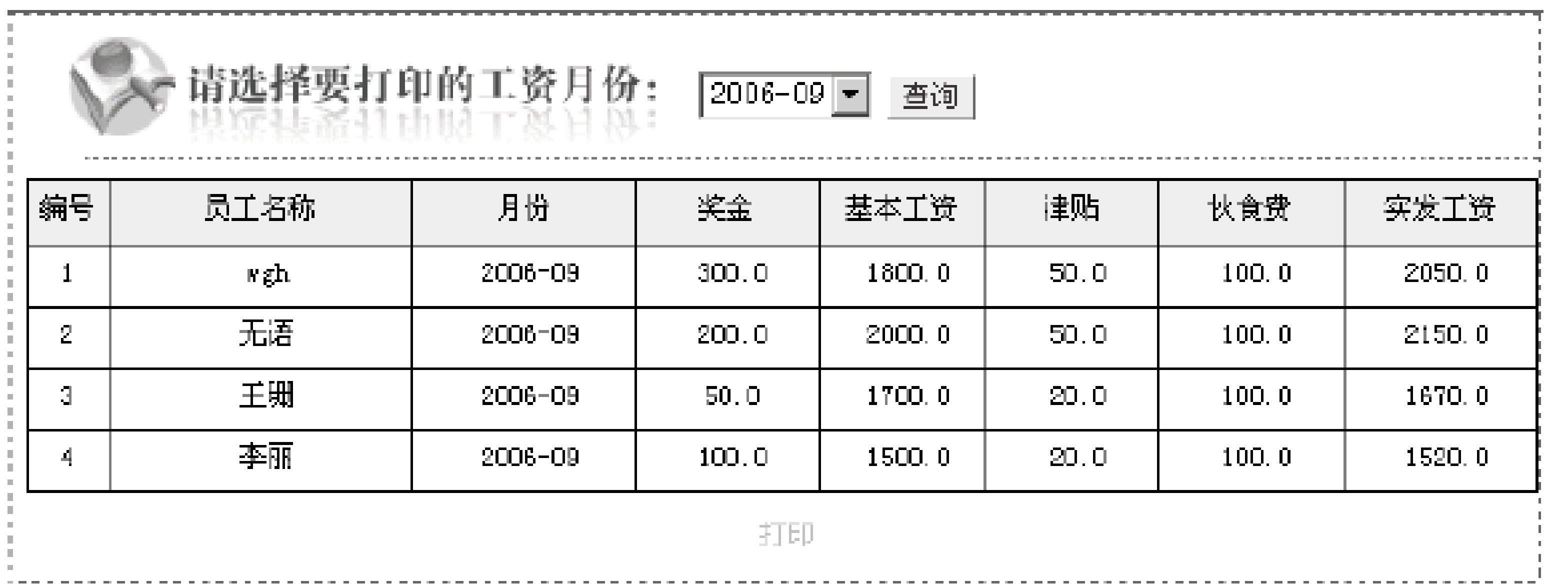


图 22.8　利用 Excel 打印工作报表



图 22.9　导出到 Excel 中的运行结果

关键技术

本实例主要是应用 JavaScript 的 ActiveXObject()构造函数创建一个 Excel.Application 对象的实例，具体语法如下：

```
var excelObj = new ActiveXObject("Excel.Application");
```

应用 Excel 对象 excelObj 中的相关方法或属性即可操作 Excel 程序。首先需要应用 excelObj 对象的 Workbooks.Add()方法创建新的工作簿，然后应用工作簿对象的 ActiveSheet 属性获取工作表，也可以应用 Worksheets()方法创建一个新的工作表。

创建工作表之后，如果想向工作表中添加内容，需要调用其 Cells()方法获取工作表中的单元格对象，Cells()方法的语法如下：

```
objSheet.Cells(row,col);
```

参数说明

❶ objSheet：表示一个工作表对象。该对象是应用工作簿对象的 ActiveSheet 属性或 Worksheets()方法创建的。

❷ row：表示工作表中单元格的行索引。索引值从 1 开始。

❸ col：表示工作表中单元格的列索引。索引值从 1 开始。

创建单元格之后，应用单元格对象 Value 属性向单元格中添加内容，应用 Font 属性设置单元格文本的样式，其中 Font 属性包含以下几个常用属性。

- ❑ Size：设置单元格字体大小。
- ❑ Name：设置单元格字体样式，如“黑体”“楷体”“宋体”等。
- ❑ Italic：设置字体为斜体。
- ❑ Bold：设置字体为粗体。
- ❑ ColorIndex：设置字体颜色，取值为整型值（如 1—黑色、2—白色、3—红色、4—绿色、5—蓝色，具体取值可以自己多做尝试）。

设计过程

（1）将显示客户信息的表格的 id 设置为 pay，因为要打印此表格中的数据，其关键代码如下：

```
<table width="643" border="0" cellspacing="1" bgcolor="#000000" id="pay">
```

（2）编写自定义 JavaScript 函数 outExcel()，用于将 Web 页面中的工资列表信息导出 Excel，并进行自动打印，其关键代码如下：

```
function outExcel(){
    var table=document.all.pay;                          //获取表格对象
    row=table.rows.length;                               //获取表格的行数
    column=table.rows(1).cells.length;                   //获取表格的列数
    var excelapp=new ActiveXObject("Excel.Application"); //创建 Excel 对象
    excelapp.visible=true;
    objBook=excelapp.Workbooks.Add();                    //添加新的工作簿
    var objSheet = objBook.ActiveSheet;                  //获取活动工作表
    title=objSheet.Range("D1").MergeArea;                //合并单元格
    title.Cells(1,1).Value ="<%=title%>";                //设置此单元格文字内容
```

```
        title.Cells(1,1).Font.Size =16;            //设置文字的大小
        for(i=1;i<row+1;i++){
                for(j=0;j<column;j++){             //通过循环向 Excel 表格的每个单元格中添加内容
                        objSheet.Cells(i+1,j+1).value=table.rows(i-1).cells(j).innerHTML.replace(" ","");
                }
        }
        objBook.SaveAs("C:/payList.xls");          //将生成的 Excel 自动保存到指定路径下
        excelapp.UserControl = true;               //自动打印
}
```

（3）通过单击“打印”超链接调用自定义 JavaScript 函数 outExcel()，其关键代码如下：

```
<a href="#" onClick="outExcel();">打印</a> >
```

秘笈心法

以上应用的 Excel 对象的相关方法，还可以实现合并 Excel 单元格操作。可以调用 Excel 对象的 Range()方法的 MergeCells 属性，在 Range()方法中需要设置合并单元格的起始位置和结束位置，例如，合并从 A 列第一格到 H 列第 6 格的区域为一个单元格，需要使用以下代码：

```
excelObj.Range("A1:H6").MergeCells=true;
```

实例 572 将页面数据导出到 Excel 并自动打印

光盘位置：光盘\MR\22\572

初级

实用指数：★★★

实例说明

在实际项目开发中，经常需要将 Web 页面中的数据导出 Excel 并实现自动打印。本实例将介绍如何将 Web 页面中的手机话费列表导出 Excel 中并保存，同时实现自动打印的功能。运行本实例，在页面中将显示手机话费列表，单击“打印”超链接，即可将手机话费列表导出到 Excel 中，并保存到“C:\”路径下，同时自动调用打印程序进行打印。程序运行结果如图 22.10 所示。



图 22.10　将页面数据导出到 Excel 并自动打印

关键技术

本实例主要通过 PrintOut()方法实现自动打印 Excel 工作表中的内容的功能。PrintOut()方法用于打印指定对象，其语法格式如下：

```
expression.PrintOut(From, To, Copies, Preview, ActivePrinter, PrintToFile, Collate, PrToFileName)
```

参数说明

❶ expression：必选参数。用于返回“Chart 对象”、“Charts 集合对象”、“Range 对象”、“Sheets 集合对象”、“Window 对象”、“Workbook 对象”、“Worksheet 对象”或“Worksheets 集合对象”中的某个对象。

❷ From：可选参数。用于指定打印的开始页号。如果省略该参数，将从起始位置开始打印。

❸ To：可选参数。用于指定打印的终止页号。如果省略该参数，将打印至最后一页。

❹ Copies：可选参数。用于指定要打印的份数。如果省略该参数，将只打印一份。

❺ Preview：可选参数。值为 true 或 false，如果为 true，则 Microsoft Excel 在打印指定对象之前先进行打印预览。如果为 false 或者省略此参数，则立即打印该对象。

❻ ActivePrinter：可选参数。用于设置活动打印机的名称。

❼ PrintToFile：可选参数。值为 true 或 false，如果为 true，则打印输出到文件。如果没有指定 PrToFileName，则 Microsoft Excel 将提示用户输入要输出文件的文件名。

❽ Collate：可选参数。当值为 true 时，将逐份打印每份副本。

❾ PrToFileName：可选参数。如果将 PrintToFile 设置为 true，则本参数指定要打印到的文件名。

设计过程

（1）将显示客户信息的表格的 id 设置为 pay，因为要打印此表格中的数据，其关键代码如下：

```
<table width="643" border="0" cellspacing="1" bgcolor="#000000" id="pay">
```

（2）编写自定义 JavaScript 函数 outExcel()，用于将 Web 页面中的手机话费列表信息导出到 Excel，并实现自动打印，其关键代码如下：

```
function outExcel(){
	var table=document.all.pay;				//获取表格
	row=table.rows.length;					//获取表格的行数
	column=table.rows(1).cells.length;		//获取表格的列数
	var excelapp=new ActiveXObject("Excel.Application");
	excelapp.visible=true;
	objBook=excelapp.Workbooks.Add();		//添加新的工作簿
	var objSheet = objBook.ActiveSheet;		//获取活动工作表
	for(i=0;i<row;i++){
		for(j=0;j<column;j++){
			objSheet.Cells(i+1,j+1).value=table.rows(i).cells(j).innerHTML.replace(" ","");
		}
	}
	objBook.SaveAs("C:/payList.xls");		//保存文件
	objSheet.Printout;						//自动打印
	excelapp.UserControl = true;
}
```

（3）通过单击“打印”超链接调用自定义 JavaScript 函数 outExcel()，其关键代码如下：

```
<a href="#" onClick="outExcel();">打印</a>
```

秘笈心法

将页面中的表格数据导出到 Excel 表格时，需要使用嵌套 for 循环，逐个找到网页的<table>表格中每个单元格的<td>内容，将其提取出来再循环添加到 Excel 中对应的单元格中。

22.4　应用 WebBrowser+CSS 套打邮寄产品单

在程序中加入打印邮寄产品单的功能，不但可以使用户非常方便地操作程序，提高工作效率，而且更能使程序适应人性化的需求。本节将通过几个实例介绍应用 WebBrowser+CSS 套打邮寄产品单的方法。

实例 573	打印汇款单 光盘位置：光盘\MR\22\573	初级 实用指数：★★★

实例说明

在开发办公自动化等系统时，可能会遇到打印汇款单的情况，这时就需要在系统中加入该功能。本实例将介绍如何在 JSP 中实现打印汇款单的功能。运行程序，在页面中将显示要打印的汇款单及与打印相关的超链接，

如图 22.11 所示，单击“打印”超链接可以打印该汇款单，单击“打印预览”超链接可以预览打印结果，如图 22.12 所示。

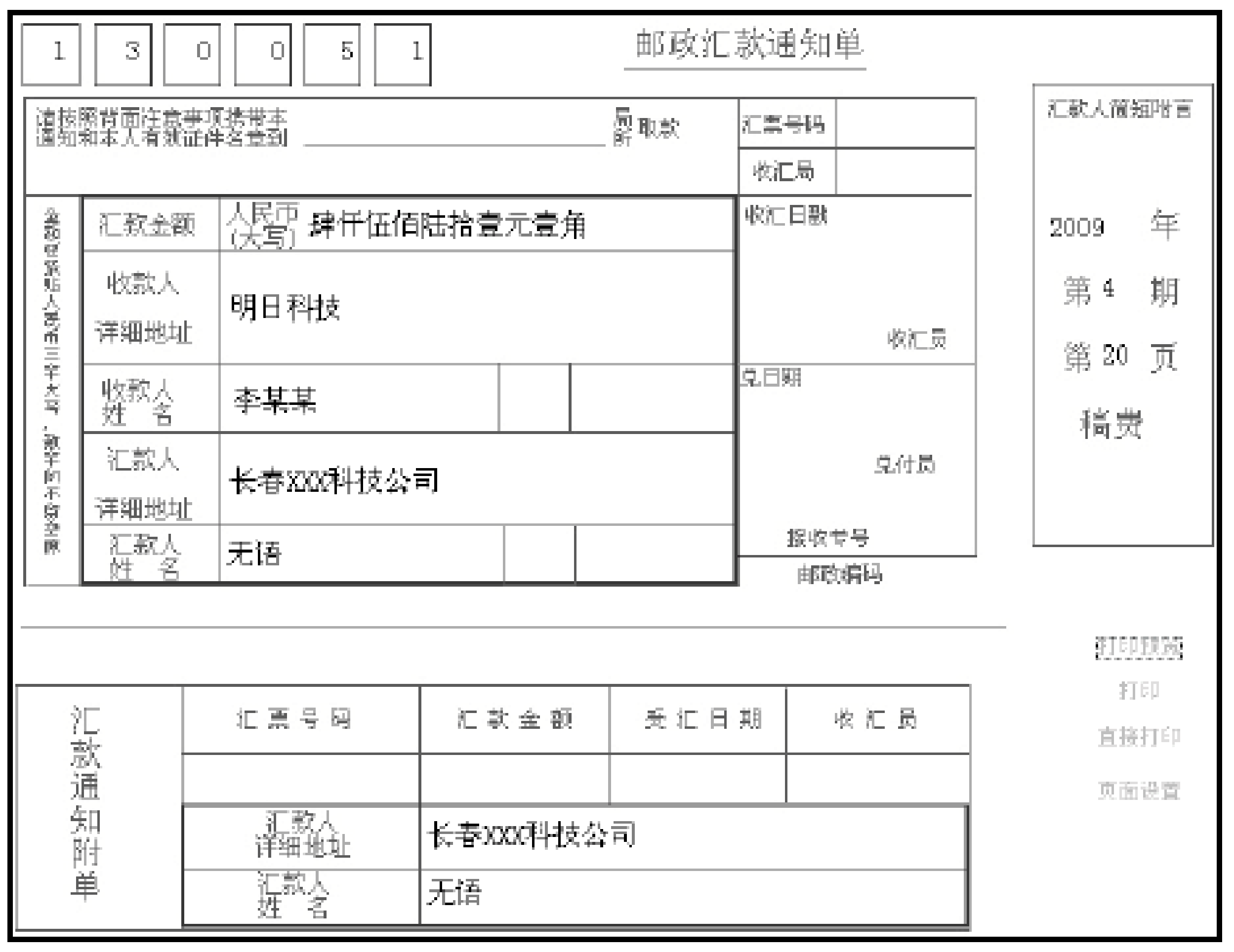


图 22.11　打印汇款单页面运行结果

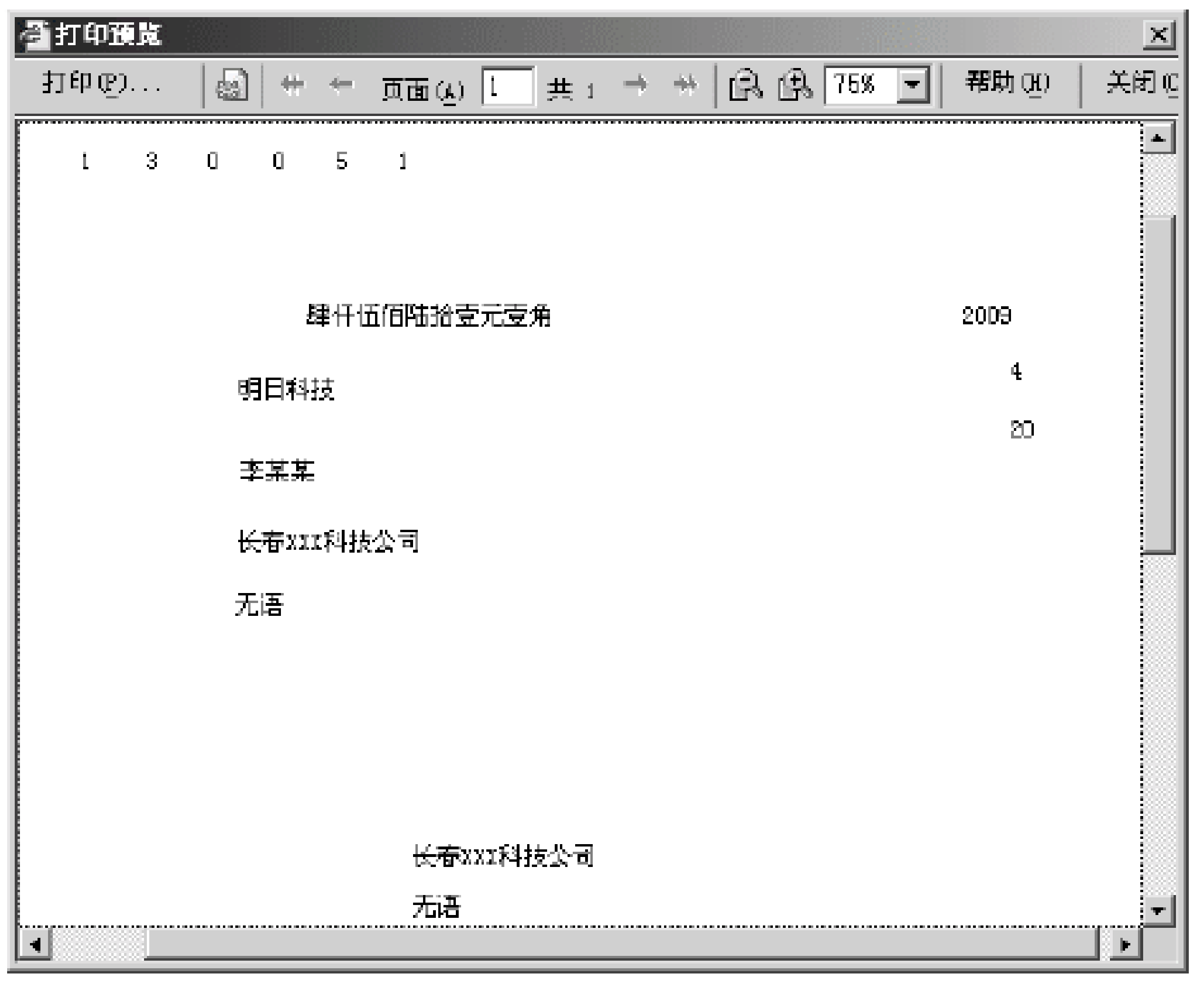


图 22.12　打印预览效果

关键技术

本实例主要通过 CSS 样式的 media 类型属性设置不打印表格背景，以及调用 WebBrowser 控件实现汇款单的打印。media 类型是 CSS 属性媒体类型，用于直接引入媒体的属性，其语法格式如下：

```
@media screen | print | projection | Braille | aural | tv | handheld | all
```

参数说明

❶ screen：指计算机屏幕。

❷ print：指用于打印机的不透明介质。

❸ projection：指用于显示的项目。

❹ Braille：盲文系统，指有触觉效果的印刷品。

❺ aural：指语音电子合成器。

❻ tv：电视类型的媒体。

❼ handheld：指手持式显示设备。

❽ all：用于所有媒体。

设计过程

（1）在页面中插入一个表格，将该表格的背景设置为空的汇款单图片，并在表格中插入新的表格，用于在指定位置显示汇款信息。

（2）在页面的指定位置填写汇款信息。

（3）在页面的相应位置加入“打印预览”“打印”“直接打印”“页面设置”超链接，其关键代码如下：

```
<div>
<table width="81" height="111" border="0" align="center" cellpadding="0" cellspacing="0">
  <tr align="center" bgcolor="#FFFFFF">
    <td colspan="3"><a href="#" onClick="document.all.WebBrowser.Execwb(7,1)">打印预览</a></td>
  </tr>
  <tr align="center" bgcolor="#FFFFFF">
    <td colspan="3"><a href="#" onClick="document.all.WebBrowser.Execwb(6,1)">打印</a></td>
  </tr>
  <tr align="center" bgcolor="#FFFFFF">
    <td colspan="3"><a href="#" onClick="document.all.WebBrowser.Execwb(6,6)">直接打印</a></td>
  </tr>
  <tr align="center" bgcolor="#FFFFFF">
    <td colspan="3"><a href="#" onClick="document.all.WebBrowser.Execwb(8,1)">页面设置</a> </td>
  </tr>
</table>
</div>
```

（4）应用 CSS 样式设置表格背景及“打印”等超链接在打印时不显示，其关键代码如下：

```
<style>
	@media print{
		div{display:none}
		td,table{
			background:display:none;
		}
	}
</style>
```

秘笈心法

本实例在超链接的 onClick 事件中，是直接编写的 JavaScript 代码。为了便于维护，避免 HTML 页面代码过于混乱，可以将此代码编写在一个 JavaScript 方法中，然后在 onClick 事件中调用 JavaScript 方法即可。

实例 574　打印信封

光盘位置：光盘\MR\22\574

初级

实用指数：★★★

实例说明

在报社或电台的日常业务中，最频繁的工作就是给订户或观众发信，如果每一封信都手工填写，不仅会降低工作效率，而且容易出错。此时通过程序控制信封的打印就可以解决该问题。运行本实例，在页面中将显示要打印的信封及与打印相关的超链接，如图 22.13 所示，单击“打印”超链接可以在信封的指定位置打印相关内容，单击“打印预览”超链接可以预览打印结果，如图 22.14 所示。



图 22.13　打印信封页面运行效果

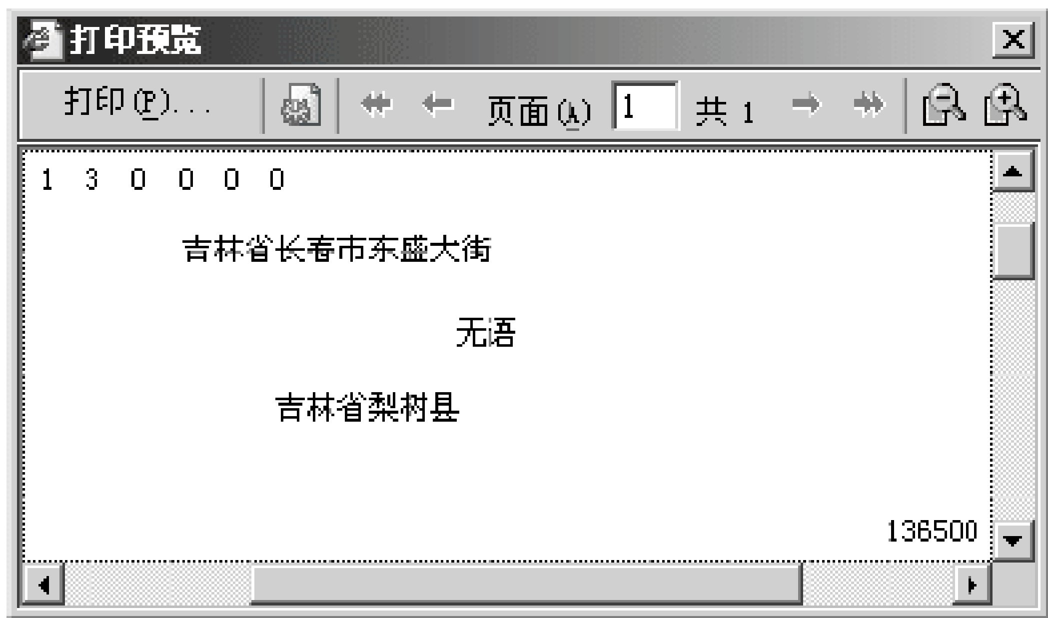


图 22.14　打印预览效果

关键技术

本实例的实现与实例 573 类似，也是通过 CSS 样式的 media 类型属性设置不打印表格背景，以及调用 WebBrowser 控件实现信封的打印。

设计过程

（1）在页面中插入一个表格，该表格的尺寸和一个标准的信封大小相同，并在表格中插入新的表格，用于在指定位置显示收信人和发信人信息。

（2）在页面的指定位置填写收信人和发信人信息。

（3）在页面的底部加入“打印预览”、“打印”、“直接打印”和“页面设置”超链接，其关键代码如下：

```
<div align="center">
	<a href="#" onClick="document.all.WebBrowser.Execwb(7,1)">打印预览</a>
	<a href="#" onClick="document.all.WebBrowser.Execwb(6,1)">打印</a>
	<a href="#" onClick="document.all.WebBrowser.Execwb(6,6)">直接打印</a>
	<a href="#" onClick="document.all.WebBrowser.Execwb(8,1)">页面设置</a>
</div>
```

（4）应用 CSS 样式控制表格背景及“打印”等超链接在打印时不显示，其关键代码如下：

```
<style>
	@media print{
```

```
            div{display:none}
            table{
                background:display:none;
            }
        }
</style>
```

秘笈心法

在 JavaScript 中，获取 HTML 元素的对象可以通过文档对象 document 的 getElementById()方法，也可以应用 document.all 的方式。其中 getElementById()方法的参数与 document.all 后的属性同样都是 HTML 元素的 id。

22.5　打印库存报表

库存管理是在信息管理系统中需要重点处理的模块，而对于库存信息的打印也是打印技术中不可缺少的环节。本节将介绍如何通过 JSP 打印各种库存报表。

实例 575	打印库存明细表 光盘位置：光盘\MR\22\575	初级 实用指数：★★★

实例说明

在开发网站或应用程序时，有时需要实现库存明细表的打印。这可以通过普通的 Web 打印实现，但是普通的 Web 打印会把控制打印的按钮或者超链接也打印出来，为了解决该问题，就需要应用 CSS 样式对打印内容进行控制。运行本实例，在页面的左下角输入每页打印的记录条数后按 Enter 键，然后单击“打印”超链接即可按用户的设置进行分页打印。程序运行结果如图 22.15 和图 22.16 所示。



图 22.15　打印库存明细表页面运行效果



图 22.16　打印预览效果

关键技术

本实例主要应用 thead 标签、tfoot 标签、CSS 样式的 media 类型和 page-break-after 属性。下面进行详细介绍。

- thead 标签：用于设置表格的表头。
- tfoot 标签：用于设置表格的表尾。
- media 类型：该类型是 CSS 属性媒体类型，详细说明请参考实例 573。
- page-break-after 属性：该属性在打印文档时发生作用，用于进行分页打印。但是对于
和<hr>对象不起作用。

page-break-after 属性的语法如下：

```
page-break-after: auto | always | avoid | left | right | null
```

参数说明

❶ page-break：打印时在样式控制的对象前后换页。

❷ after：设置对象后出现页分割符。设置为 always 时，始终在对象之后插入页分割符。

❸ auto：需要在对象之后插入页分割符时插入。

❹ always：始终在对象之后插入页分割符。

❺ avoid：未支持。避免在对象后面插入分割符。

❻ left：未支持。在对象后面插入页分割符，直到它到达一个空白的左页边。

❼ right：未支持。在对象后面插入页分割符，直到它到达一个空白的右页边。

❽ null：空白字符串。取消了分割符设置。

设计过程

（1）在要打印的页面中添加用于显示库存明细信息的表格，并设置好表头、表尾及打印分页，其关键代码如下：

```
<table width="98%" border="1" id="pay" style="border-bottom-style:none;">
<thead style="display:table-header-group;font-weight:bold">
        <tr>
          <td width="35" height="27" align="center" bgcolor="#efefef">编号</td>
          <td width="140" align="center" bgcolor="#efefef">商品名称</td>
          <td width="103" align="center" bgcolor="#efefef">产地</td>
          <td width="82" align="center" bgcolor="#efefef">规格</td>
          <td width="61" align="center" bgcolor="#efefef">计量单位</td>
          <td width="75" align="center" bgcolor="#efefef">单价</td>
          <td width="65" align="center" bgcolor="#efefef">库存数量</td>
        </tr>
</thead>
<%
int i=1;              //商品编号
String spname="";   //商品名称
String cd="";         //产地
String gg="";         //规格
String dw="";         //单位
float dj=0f;          //单价
int kcsl=0;           //库存数量
String rsRow=request.getParameter("rsRow");   //每页打印的条数
int intRsRow=10;   //默认每页打印 10 条
if(rsRow!=null && !rsRow.equals("")){
        intRsRow=Integer.parseInt(rsRow);
}
try{
   while(rs.next()){
        spname=rs.getString(2);
        cd=rs.getString(3);
        gg=rs.getString(4);
        dw=rs.getString(5);
        dj=rs.getFloat(6);
        kcsl=rs.getInt(7);
%>
<tr<%if(i%intRsRow==0){%> style="page-break-after:always;"<%}%>>
   <td height="25" align="center" bgcolor="#FFFFFF"><%=i%></td>
   <td align="center" bgcolor="#FFFFFF"> <%=spname%></td>
   <td align="center" bgcolor="#FFFFFF"> <%=cd%></td>
   <td align="center" bgcolor="#FFFFFF"> <%=gg%></td>
   <td align="center" bgcolor="#FFFFFF"> <%=dw%></td>
   <td align="center" bgcolor="#FFFFFF"> <%=dj%></td>
   <td align="center" bgcolor="#FFFFFF"> <%=kcsl%></td>
</tr>
<%
        i++;
}
```

```
    %>
    <tfoot style="display:table-footer-group; border:none;"><tr><td></td></tr></tfoot>
    </table>
<%
    }catch(Exception e){
        System.out.println(e.getMessage());
    }
%>
```

（2）控制“打印”超链接及每页打印记录条数的表单，并设置 CSS 样式在打印时不显示，其关键代码如下：

```
<style>
        @media print{
                div{display:none}
        }
</style>
<div align="center">
        <table width="95%" height="27" border="0" cellspacing="0" cellpadding="0">
            <tr>
                <td width="47%">每页打印<input name="rsRow" type="text" value="<%=intRsRow%>" size="3" onKeyDown="enter()">条记录
                </td>
                <td width="53%" align="right"><a href="#" onClick="window.print();">打印</a></td>
            </tr>
    </table>
</div>
```

秘笈心法

在进行 Web 打印时，可以通过以下操作控制是否打印背景颜色和图像。在 IE 窗口中，选择“工具”→“Internet 选项”命令，在弹出的“Internet 选项”对话框中打开“高级”选项卡，查看“设置”列表中“打印背景颜色和图像”前面的复选框是否被选中，如果选中，则代表打印背景颜色和图像，否则不打印背景颜色和图像。

实例 576	打印库存盘点报表 光盘位置：光盘\MR\22\576	初级 实用指数：★★★

实例说明

在开发网络应用程序时，经常需要打印库存盘点报表。本实例将介绍如何在 JSP 中实现打印库存盘点报表的功能。运行本实例，在页面的左下角输入每页打印的记录条数后按 Enter 键，然后单击“打印”超链接即可按用户的设置进行分页打印。程序运行结果如图 22.17 和图 22.18 所示。

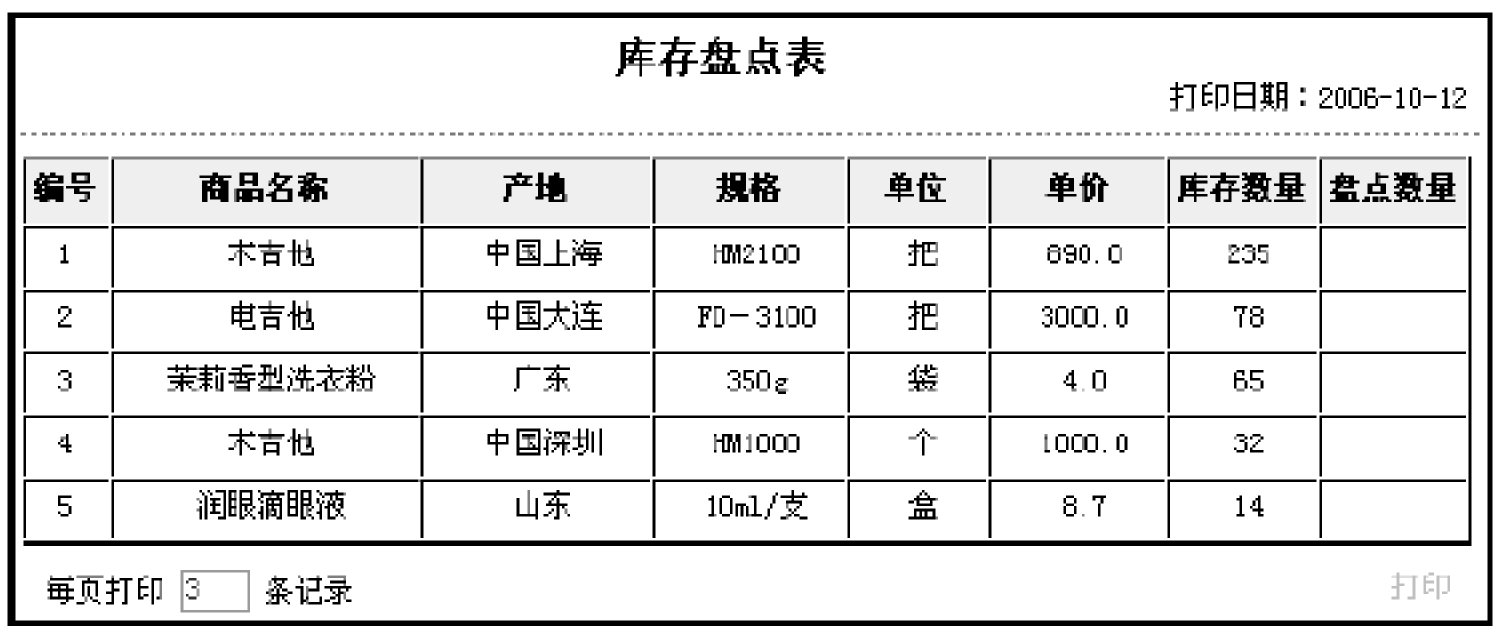


图 22.17　打印库存盘点报表预览效果



图 22.18　打印预览效果

关键技术

本实例主要采用 HTML 标记和 CSS 样式控制页面中打印的内容，并为每一页设置表头、表尾，然后通过 window.print()语句实现打印库存盘点报表的功能。

设计过程

（1）在要打印的页面中添加用于显示库存盘点报表内容的表格，并设置好表头、表尾及打印分页，其关键

代码如下：

```
<table width="98%" id="pay" style="border-bottom-style:none;">
 <thead style="display:table-header-group;font-weight:bold;">
        <tr>
          <td width="35" height="27" align="center" bgcolor="#efefef">编号</td>
          <td width="140" align="center" bgcolor="#efefef">商品名称</td>
          <td width="103" align="center" bgcolor="#efefef">产地</td>
          <td width="82" align="center" bgcolor="#efefef">规格</td>
          <td width="61" align="center" bgcolor="#efefef">单位</td>
          <td width="75" align="center" bgcolor="#efefef">单价</td>
          <td width="65" align="center" bgcolor="#efefef">库存数量</td>
          <td width="65" align="center" bgcolor="#efefef">盘点数量</td>
        </tr>
 </thead>
      ……//省略了从数据流表中获取信息的代码
      <tr<%if(i%intRsRow==0){%> style="page-break-after:always;"<%}%>>
        <td height="25" align="center" bgcolor="#FFFFFF"><%=i%></td>
        ……
</tr>
      ……
      <tfoot style="display:table-footer-group; border:none;"><tr><td></td></tr></tfoot>
  </table>
```

（2）控制“打印”超链接及每页打印记录条数的表单，在打印时不显示，其关键代码如下：

```
<style>
        @media print{
                div{display:none}
        }
</style>
<div align="center"><table width="95%" height="27" border="0" cellspacing="0" cellpadding="0">
                <tr>
                  <td width="47%">每页打印<input name="rsRow" type="text" value="<%=intRsRow%>" size="3" onKeyDown="enter()">条记录</td>
                  <td width="53%" align="right"><a href="#" onClick="window.print();">打印</a></td>
                </tr>
      </table>
</div>
```

秘笈心法

在实现本实例时，需要注意的是在打印报表时，由于盘点数量需要手工输入，所以在打印时需要留出相应的位置。

实例 577	打印库存汇总报表 光盘位置：光盘\MR\22\577	初级 实用指数：★★★

实例说明

在开发网络应用程序时，经常需要打印库存汇总报表。本实例将介绍如何在 JSP 中实现打印库存汇总报表的功能。运行本实例，在页面的左下角输入每页打印的记录条数后按 Enter 键，再单击“打印”超链接即可按用户的设置分页打印库存汇总报表。在打印时，在每一页的最后一行将打印本页的合计金额，同时在最后一页还将打印库存总金额。程序运行结果如图 22.19 和图 22.20 所示。

关键技术

实现本实例，最关键的步骤是计算本页合计金额并控制其显示位置。下面给出实现该功能的基本思路：

（1）计算每种商品的库存金额，结果保留两位小数。

（2）通过 While 循环累加每种商品的库存金额 total1。

（3）当 i（当前记录数）可以被“每页显示的记录数”整除，并且不是最后一条记录时，显示本页合计金额，并将 total1 清零。

（4）显示最后一页中的本页合计金额。



图 22.19　打印库存汇总报表页面运行效果



图 22.20　打印预览效果

设计过程

（1）在要打印的页面中添加用于显示库存信息的表格，并设置好表头及表尾。

（2）计算本页合计金额并控制其显示的位置，同时控制打印分页，其关键代码如下：

```
<%
  //计算本页合计金额
  int intRsRow=10;
  Object[] je = new Object[1];
  String rsRow=request.getParameter("rsRow");
  float total1=0f;
  float fje=0f;
  rs.last();
  int rsTotalRow=rs.getRow();                                  //获取记录总数
  rs.first();
  do{
      je[0] = new Float(dj*kcsl);
      fje=Float.valueOf(String.format("%.2f", je)).floatValue();
      total1=total1+fje;
    %>
   //显示本页合计金额并分页
<%if(i%intRsRow==0 &&i!=rsTotalRow){%>
      <tr style="page-break-after:always;">
        <td align="right" height="25" colspan="8" bgcolor="#efefef"> 本页合计金额：<%=total1 %>(元)</td>
      </tr>
    <%total1=0;}%>
<%i++;
}while(rs.next());%>
        //显示最后一页中的本页合计金额
      <tr>
        <td align="right" height="25" colspan="8" bgcolor="#efefef"> 本页合计金额：<%=total1 %>(元)</td>
      </tr>
```

（3）计算库存总金额并显示，其关键代码如下：

```
<%
  float total=0f;
  float fje=0f;
  do{
       je[0] = new Float(dj*kcsl);                            //计算单件商品的库存金额
      fje=Float.valueOf(String.format("%.2f", je)).floatValue();
      total=total+fje;                                       //计算库存总金额
%>
<table width="98%" height="23" border="0" cellpadding="0" cellspacing="0">
  <tr><td align="right"> 库存总金额：<%=total%>(元)</td></tr>
</table>
```

```
<%i++;
}while(rs.next());%>
```

（4）控制“打印”超链接及每页打印记录条数的表单在打印时不显示，其关键代码如下：

```
<style>
@media print{
div{display:none}

}
</style>
<div align="center">
    <table width="95%" height="27" border="0" cellspacing="0" cellpadding="0">
            <tr><td width="47%">每页打印<input name="rsRow" type="text" value="<%=intRsRow%>" size="3" onKeyDown="enter()">条记录
</td>
              <td width="53%" align="right"><a href="#" onClick="window.print();">打印</a></td>
            </tr>
      </table>
</div>
```

秘笈心法

在本实例中，每一页的汇总金额是通过循环数据库返回的结果集计算出来的，在循环中会累加金额信息，当 i（当前记录数）可以被“每页显示的记录数”整除，并且不是最后一条记录时，显示本页合计金额，并将 total1 清零。

实例 578　打印指定条件的库存报表

光盘位置：光盘\MR\22\578

初级

实用指数：★★★

实例说明

在开发 Web 应用程序时，经常会遇到打印指定条件的库存报表的情况，这就需要在程序中加入打印指定条件的库存报表的功能。运行程序，输入如图 22.21 所示的条件，单击“打印”超链接，即可根据条件打印所需的信息，如图 22.22 所示。



图 22.21　在页面中输入查询条件查询库存报表



图 22.22　打印预览效果

关键技术

实现本实例，首先要进行条件查询，并将查询结果显示在指定框架中，然后应用打印指定框架的方法实现打印指定条件的库存报表的功能。打印指定框架的方法的语法格式如下：

```
parent.mainFrame.focus;window.print();
```

参数说明

mainFrame：表示要打印框架的名称。

设计过程

（1）创建 index.jsp 页，添加上下分栏的框架页，该框架页的顶部为查询条件页面 top.jsp，底部为显示查询结果页面 bottom.jsp，其关键代码如下：

```
<frameset rows="80,*" frameborder="no" border="0" framespacing="0">
  <frame src="top.jsp" name="topFrame" scrolling="No" noresize="noresize" id="topFrame" />
```

```
  <frame src="bottom.jsp" name="mainFrame" id="mainFrame" />
</frameset>
<noframes><body></body></noframes>
```

（2）在 bottom.jsp 页中查询库存报表信息，并设置每页打印的记录数和控制分页。具体方法可参见实例 575。

（3）在 bottom.jsp 页中添加“打印”超链接，应用打印指定框架的方法实现打印指定条件的库存报表的功能，其关键代码如下：

```
<a href="#" onClick="parent.mainFrame.focus;window.print();">打印</a>
```

秘笈心法

在超链接中，调用父页面 Frame 框架对象的 focus 属性，含义是使框架获得焦点，在执行打印时，直接打印焦点中的内容。

22.6 高 级 报 表

在实际项目开发过程中，有时需要实现比较复杂的报表，如主从报表或分栏报表。本节将通过两个实例介绍如何应用 iReport+JasperReport 生成主从报表和分栏报表。

实例 579	应用 iReport+JasperReport 生成主从报表 光盘位置：光盘\MR\22\579	高级 实用指数：★★★

实例说明

主从报表又称为复合报表，即报表本身又包含一张报表，这张报表被称为第一张报表的子报表，实质上子报表也是一张真正的报表，子报表也会在使用之前首先进行编译操作，父报表与子报表的关系可以是从属关系也可以是并列关系。本实例将介绍如何在 JSP 中应用 iReport+JasperReport 生成主从报表。运行程序，将显示如图 22.23 所示的页面，在该页面中，单击“订单报表”超链接，将生成保存客户订单信息的 PDF 格式的主从报表，并且通过 PDF 阅读器自动打开，如图 22.24 所示。



图 22.23 应用 iReport+JasperReport 生成主从报表主页面



图 22.24 通过 PDF 阅读器打开生成的主从报表

关键技术

本实例主要应用 iReport 与 JasperReport 组件实现。下面将对 iReport 和 JasperReport 组件进行详细介绍。

1. iReport 组件

iReport 是制作报表工具，它具有良好的人性化界面，并拥有高性能，使用该软件程序员可以随意设计报表，使得报表不再千篇一律，并使得分组、分栏统计变得非常便捷。

iReport 是开源项目，可以到 http://jasperforge.org/plugins/project/project_home.php?group_id=83 网站下载最新版本的 iReport。当前最新版本为 3.1.4，下载后的文件名称为 iReport-nb-3.1.4-windows-installer.exe，双击该文件即可安装 iReport。安装完成后，即可运行该程序，iReport 的主界面如图 22.25 所示。

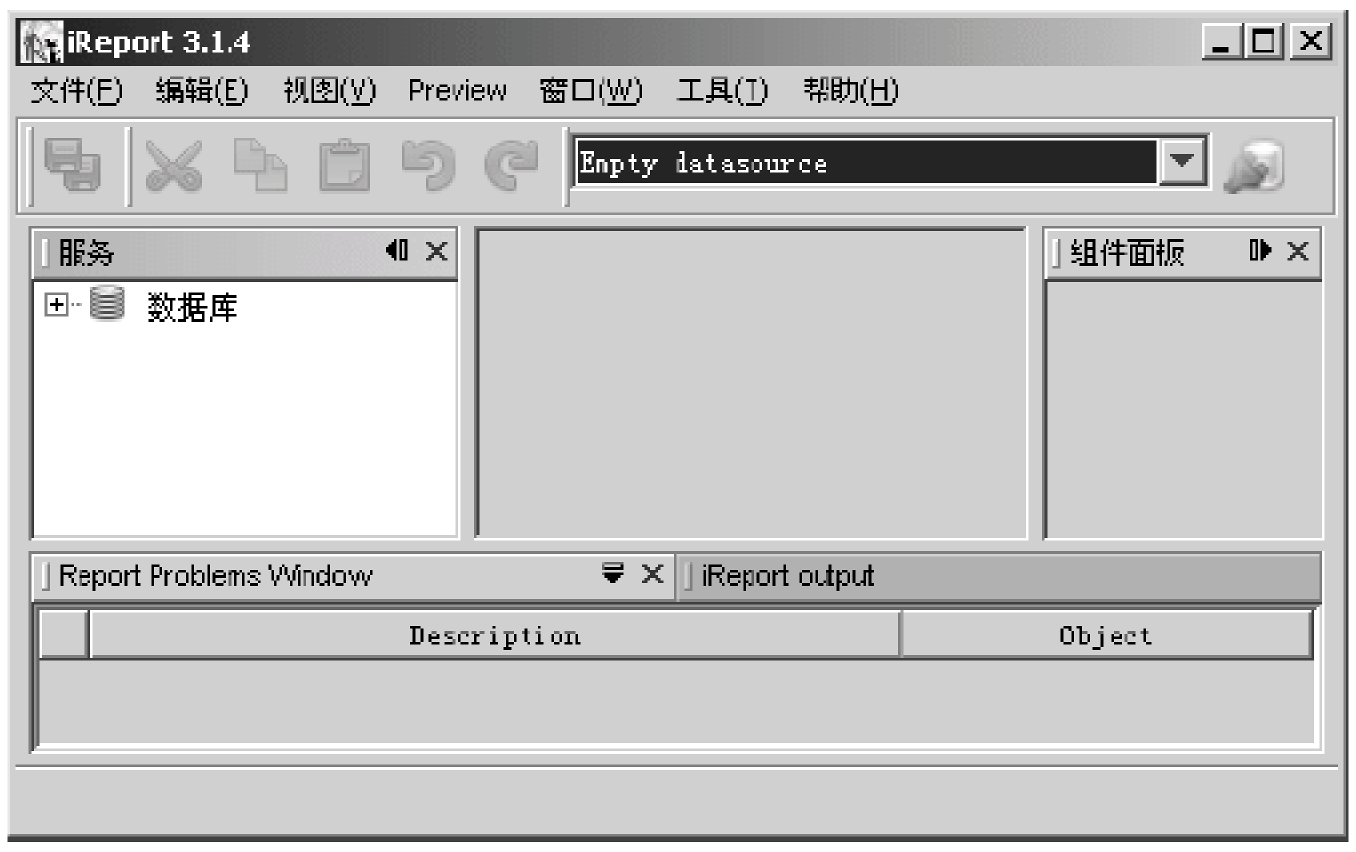


图 22.25　iReport 3.1.4 的主界面

2. JasperReport 组件

JasperReport 组件是一个强大的报表产生工具，使用它可以将在 iReport 中制作的报表与 JSP 页面相联系。JasperReport 可以生成多种类型的报表，如 PDF、HTML 或 XML 等，并支持 CSV、XLS 等格式的报表。

JasperReport 组件也是开源项目，可以到 http://jasperforge.org/plugins/project/project_home.php?group_id=102 网站下载最新版本的 JasperReport 组件。当前最新版本为 3.1.4，下载后的文件名称为 jasperreports-3.1.4.jar。

设计过程

（1）运行 iReport，选择主菜单中的“文件”→New→Empty report 命令，打开 New 对话框，在 Report name 文本框中输入报表名称 zhucongReport，在 Location 文本框中输入该报表文件保存的路径，这里为 E:\09\zhucongReport，当然也可以通过单击 Browse 按钮选择文件保存的位置，如图 22.26 所示。

（2）单击“完成”按钮，完成报表文件的创建。

（3）新建报表完成后，需要为该报表配置相应的数据源，也就是指定该报表需要访问的数据库。

首先将所使用数据库的驱动包添加到 iReport 的 Classpath 中。选择主菜单中的“工具”→“选项”命令，在打开的“选项”对话框中，选择 iReport 节点下的 Classpath 选项卡，单击 Add JAR 按钮，在打开的添加 JAR 对话框中选择数据库驱动，这里为 SQL Server 2000 数据库的驱动包，单击“打开”按钮，即可将数据库驱动包添加到 Classpath 下，如图 22.27 所示。单击“确定”按钮，完成数据库驱动包的添加。

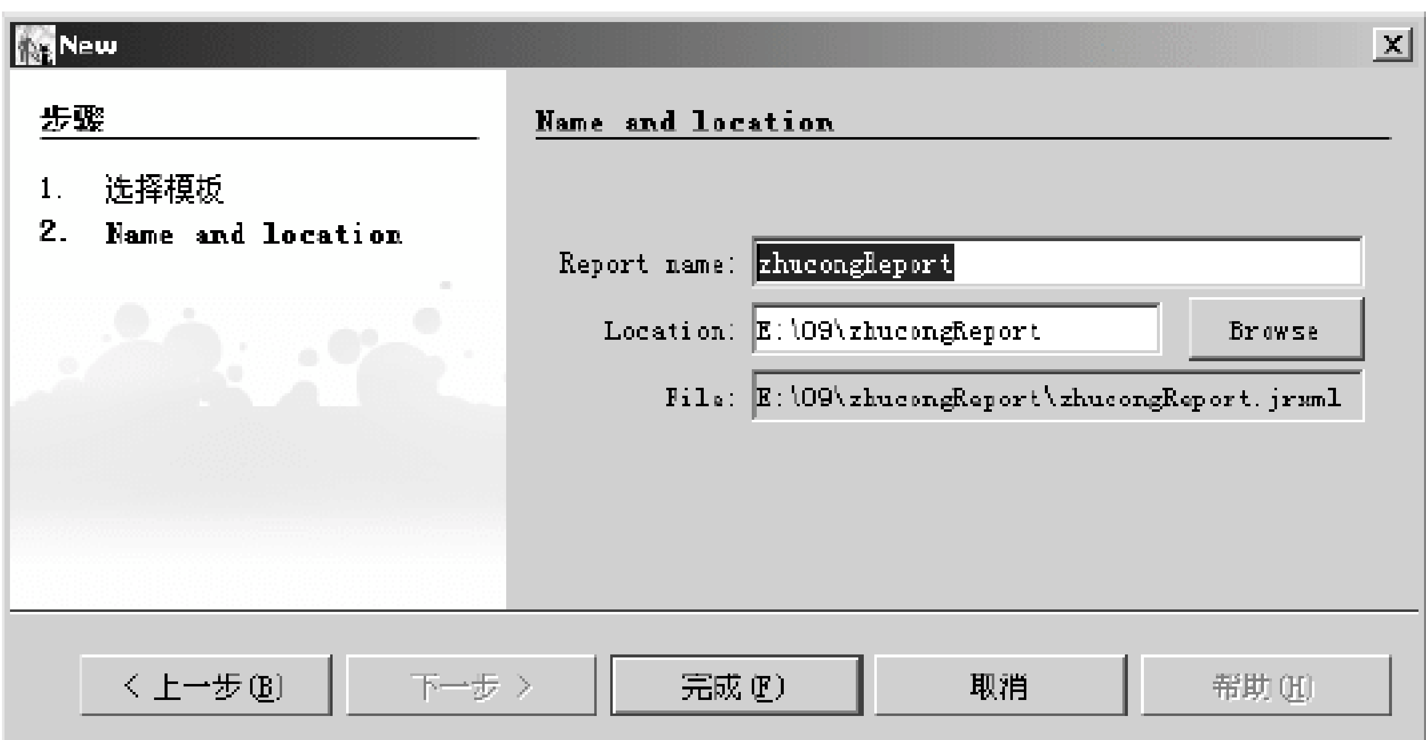


图 22.26　New 对话框

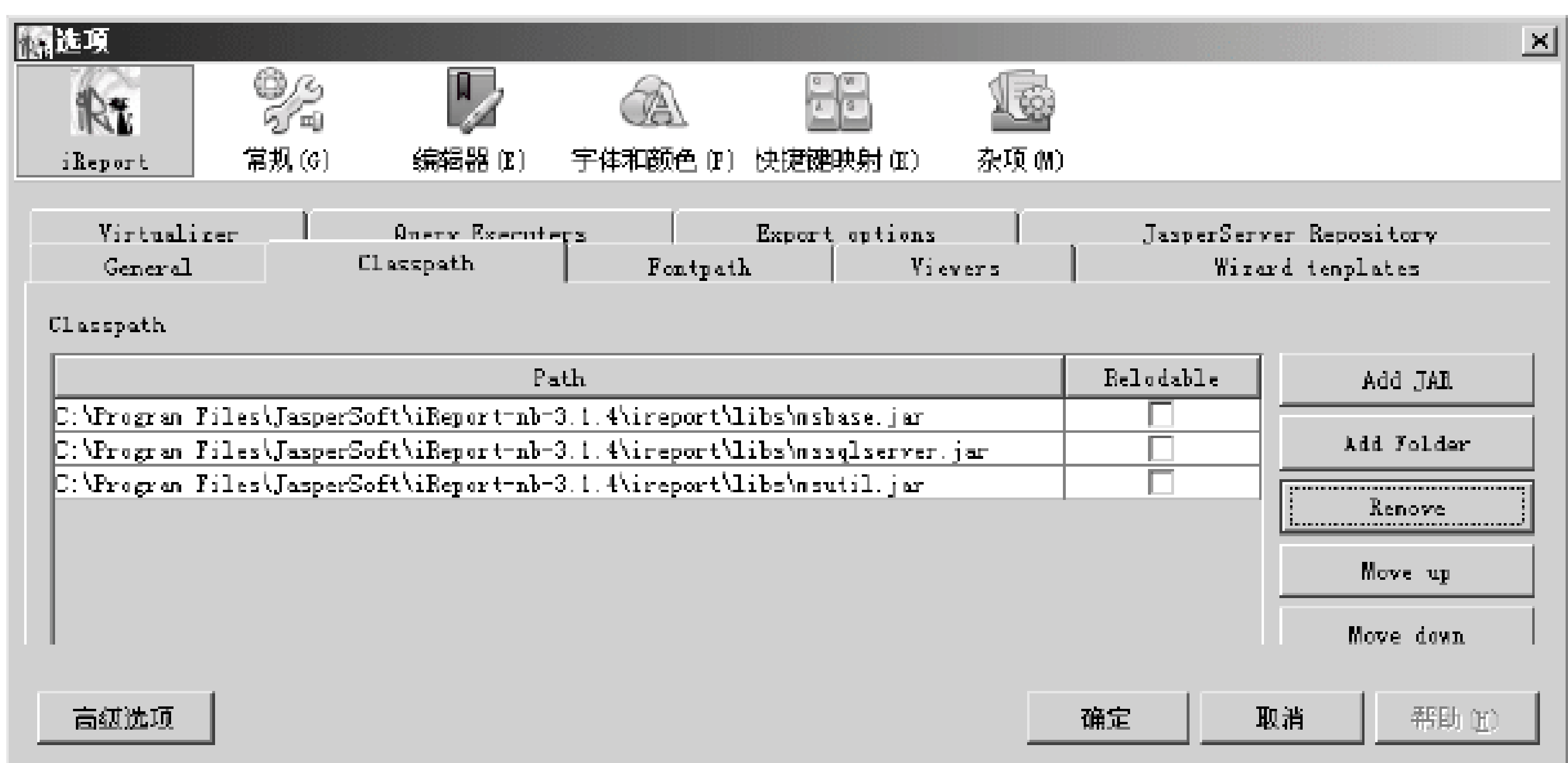


图 22.27　“选项”对话框的 Classpath 选项卡

然后就可以创建数据源。选择主菜单中的“工具”→Report Datasource 命令，将打开如图 22.28 所示的 Connections/Datasources 对话框。

在该对话框中单击 New 按钮，将打开 Datasource 对话框，在该对话框中选中 Database JDBC Connection 节点，单击 Next 按钮，将打开 Database JDBC connection 对话框，在该对话框的 Name 文本框中输入数据源名，在 JDBC Driver 下拉列表框中选择数据库驱动名，这里为 MS SQL Server (2000) (com.microsoft.jdbc.sqlserver.SQLServerDriver)，修改 JDBC URL 文本框中生成的 JDBC URL，在 Server Address 文本框中输入服务器地址，这里为 localhost，在 Database 文本框中输入数据库名，这里为 db_database09，在 Username 文本框中输入用户名，这里为 sa，密码为空，选中 Save password 复选框，保存密码，如图 22.29 所示。

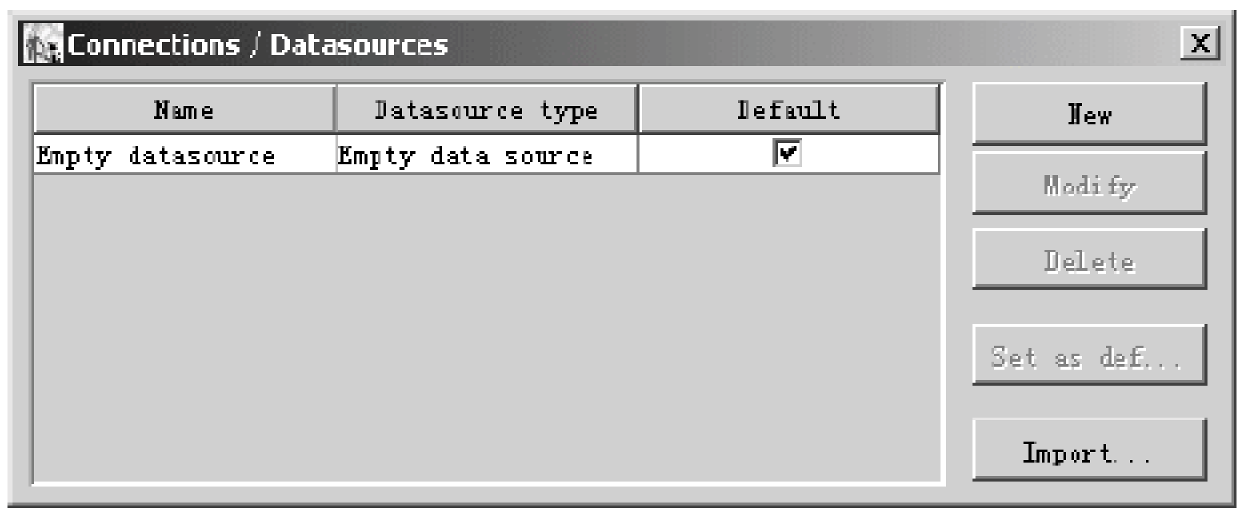


图 22.28　Connections/Datasources 对话框

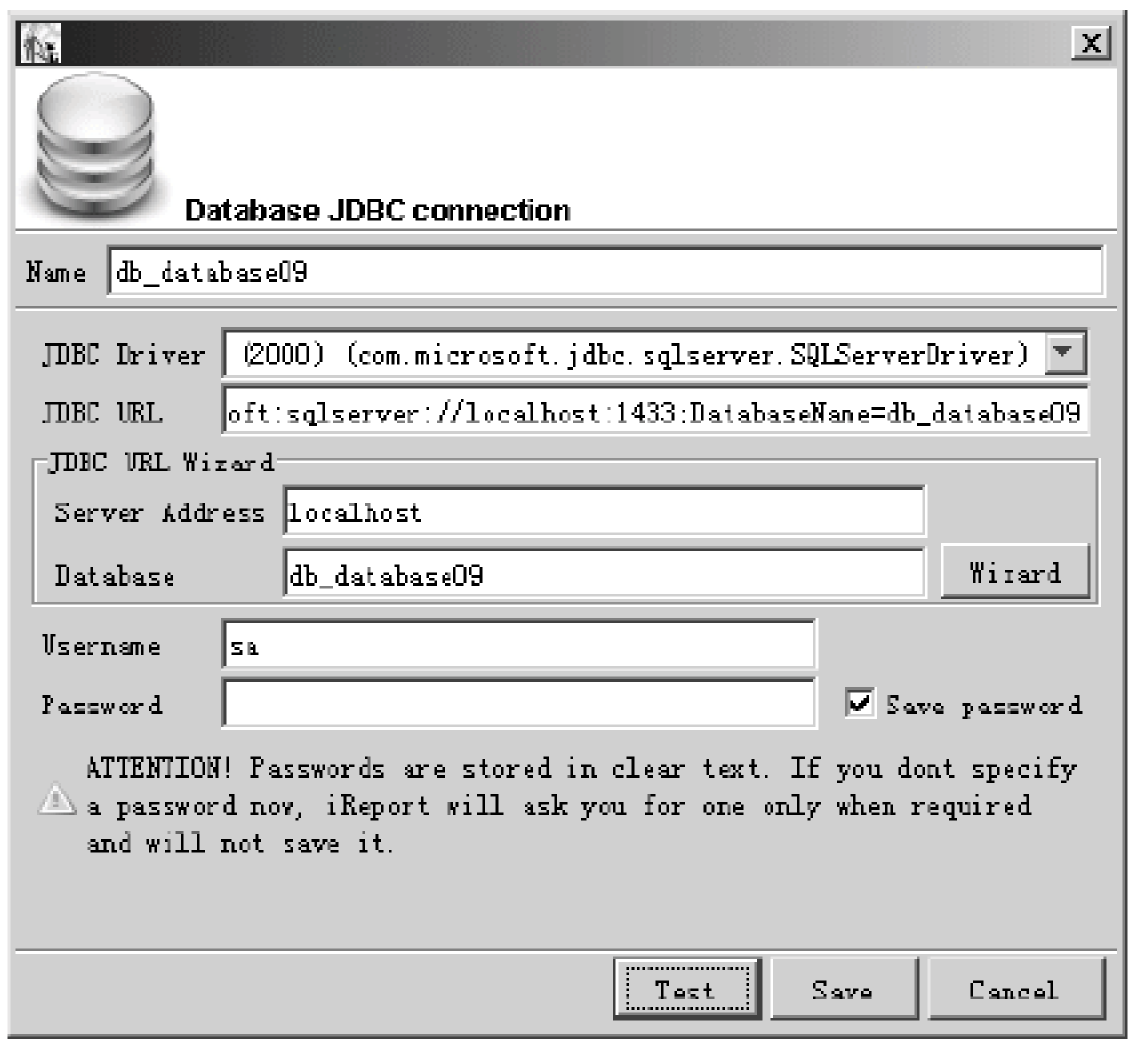


图 22.29　Database JDBC connection 对话框

单击 Test 按钮，测试数据源连接，如果成功，将显示 Connection test successful!对话框，否则将显示错误提示对话框，读者可以根据提示信息进行修改，直到提示连接成功。单击 Save 按钮，保存该数据源。这时该数据源将自动被选中。

（4）当数据源设置完成后，可以使用 SQL 语句指定报表中需要查询的数据。

在 iReport 主界面左侧的 Report Inspector 面板中，选中 report name 节点，并在该节点上单击鼠标右键，在弹出的快捷菜单中选择 Edit Query 命令，将打开 Report query 对话框，在该对话框的文本域中输入以下 SQL 语句：

```
SELECT * FROM tb_customerInfo
```

在该对话框下方的字段表格中将显示数据表中所有字段的相关信息，如图 22.30 所示。

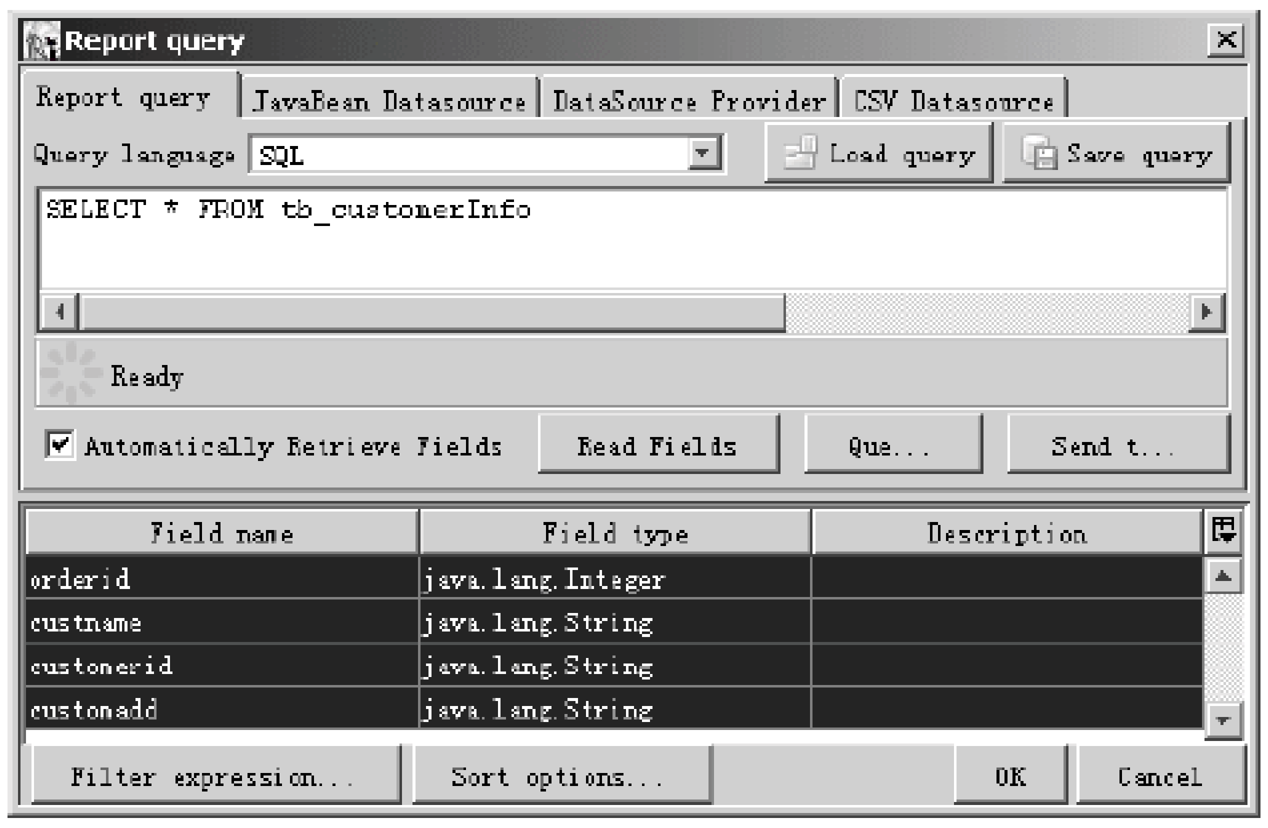


图 22.30　Report query 对话框

设置完成报表查询的 SQL 语句，单击 OK 按钮即可。这时，在 Report Inspector 面板的 Fields 节点中将显示该查询结果中包括的全部字段。

（5）按照步骤（3）的方法再向 iReport 的 Classpath 中添加 jasperreports-3.1.4.jar 和 iTextAsian.jar 包，其中 iTextAsian.jar 用于解决生成报表不支持中文的问题。

（6）在报表设计器中，将 Title 区域、Column Header 区域、Column Footer 区域和 Summary 区域的 Band height

属性设置为 0；将 Page Header 区域的 Band height 属性设置为 52；将 Detail 区域的 Band height 属性设置为 130；将 Page Footer 区域的 Band height 属性设置为 31。

（7）在“组件面板”中拖动 Image 图标到报表设计器的 Page Header 区域中，这时将打开选择图片文件对话框，这里先不选择要放置的图片，单击“取消”按钮，选中刚刚添加的图片对象，在属性面板中将 Left 属性设置为-19、Top 属性设置为-20、width 属性设置为 593、height 属性设置为 71、Express Class 设置为 java.lang.String。

（8）为了使图片的路径在每次报表编译时都被确定，可以将图片的路径设置为参数的形式。在 Report Inspector 面板的 Parameters 节点上单击鼠标右键，在弹出的快捷菜单中选择“添加 Parameter”命令，创建一个参数，在属性面板中，设置该参数的 name 属性为 url，Parameter Class 为 java.lang.String，其他采用默认。参数设置完成后，还需要将图片的 Image Expression 属性设置为$P{url}。

（9）按照步骤（7）和步骤（8）介绍的方法在 Page Footer 区域中添加一个页脚图片。

（10）在 Detail 区域中添加 3 个 static Text 和 3 个 Text Field，并设置 static Text 的 Text 属性为要显示的文字；设置 Text Field 的 Text Field Expression 属性为代表要显示字段的表达式。设置完成的效果如图 22.31 所示。

为了让报表支持中文，还需要将这 6 个组件的 Pdf Font name 属性设置为 STSong-Light，将 Pdf Encoding 属性设置为 UniGB-UCS2-H (Chinese Simplified)，如果要显示文字为纵向排列，则设置为 UniGB-UCS2-V (Chinese Simplified)。

（11）创建子报表。

选择主菜单中的“文件”→New→Empty report 命令，在打开的 New 对话框的 Report name 文本框中输入报表名称 subReport，在 Location 文本框中输入该报表文件保存的路径，这里为 E:\09\zhucongReport，当然也可以通过单击 Browse 按钮选择文件保存的位置，单击“完成”按钮创建一个空的报表。

在该报表中，将 Column Header 区域的 Band Height 属性设置为 31；将 Detail 区域的 Band Height 属性设置为 29；其他区域的 Band Height 属性设置为 0。

创建名称为 custname 的参数，将该参数的 Parameter Class 属性设置为 java.lang.String，将 Default Value Expression 属性设置为""，其他采用默认，如图 22.32 所示。创建该参数的目的是将父报表的 tb_customerInfo 表中的 custname 字段的值作为参数传入子报表，子报表根据 SQL 语句检索 tb_orderinfo 表中对应的数据。



图 22.31　放置字段与字段内容



图 22.32　custname 参数的属性设置

按照步骤（4）中介绍的方法指定报表中需要查询的数据。具体的 SQL 语句如下：

```
select *,num*price as zj from tb_orderinfo where custname=$P{custname}
```

在报表设计器的 Column Header 和 detail 区域中，将字段名称与信息放置在合适的位置，并通过 Line 组件绘制表格。设置完成的效果如图 22.33 所示。

注意：在绘制表格或添加字段信息时，不能有重叠的组件，否则在生成的报表中将不显示。

至此，子报表已经设计完成。选择主菜单中的 preview→PDF preview 命令，设置预览报表的格式为 PDF，单击 Preview 视图，预览报表，这时 iReport 会弹出提示对话框，要求输入参数 custname 的值，这里输入“小兰”，单击 OK 按钮，即可预览该报表，如图 22.34 所示。



图 22.33　创建子报表数据区域



图 22.34　预览子报表

（12）返回到主报表中，在主报表中调用子报表。

将组件面板中的 Subreport 组件拖动到 Detail 区域，这时将打开 Subreport wizard 对话框，在该对话框中选中 Use an existing report 单选按钮，单击 Browse 按钮，选择创建完成的子报表，如图 22.35 所示。

单击“下一步”按钮，切换到子报表向导第 2 步，指定所用连接，这里采用默认设置，如图 22.36 所示。

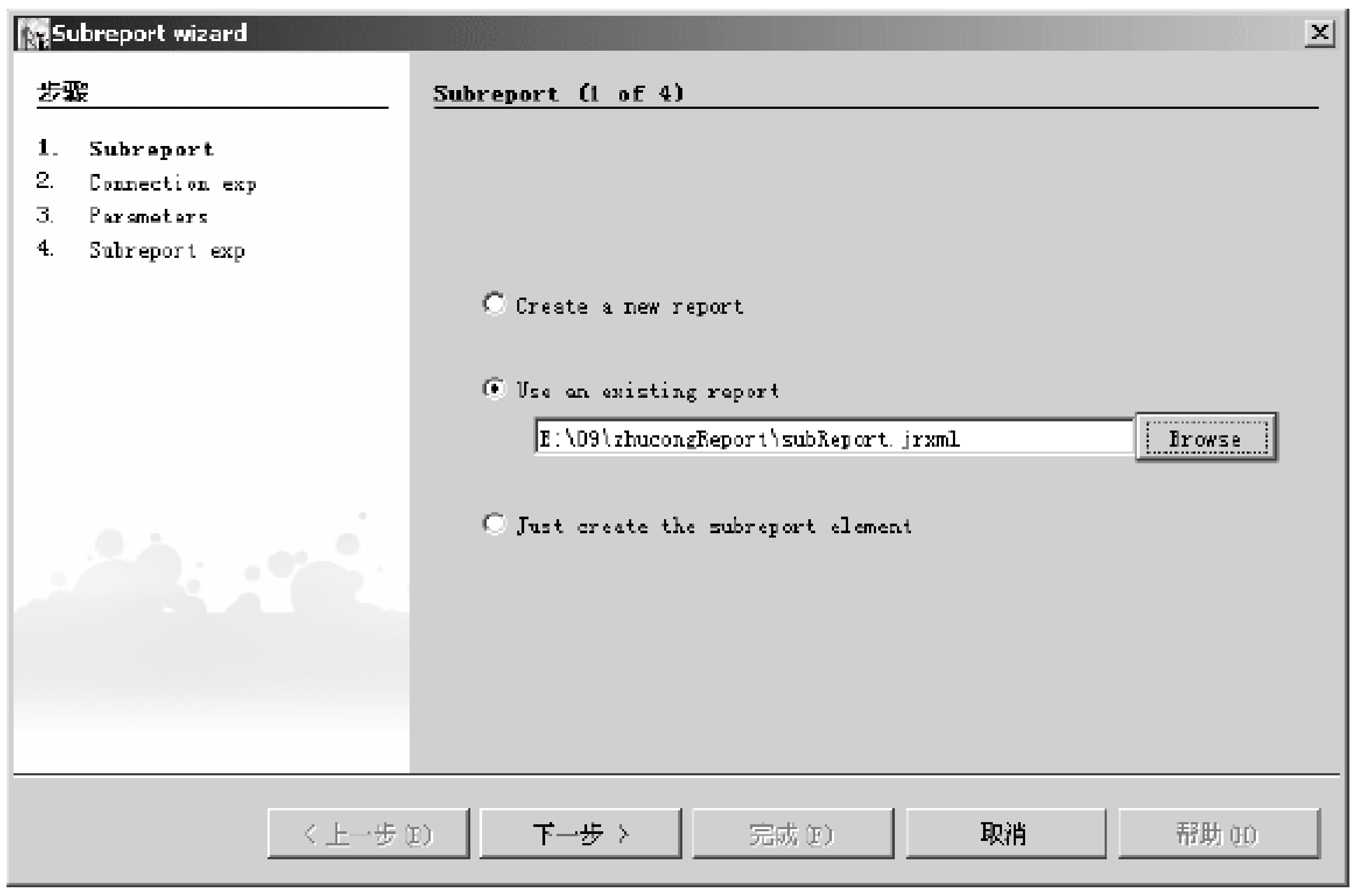


图 22.35　选择子报表

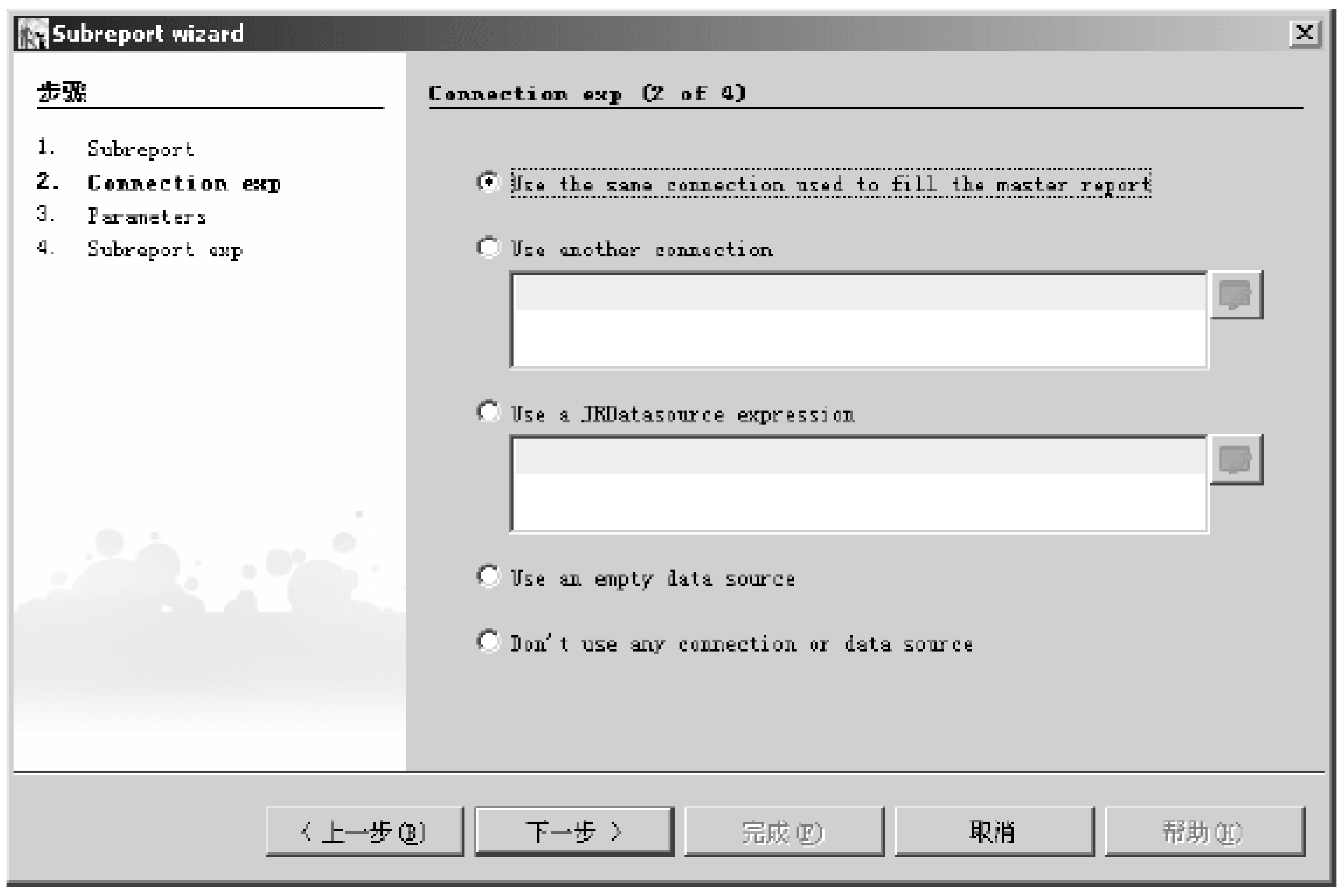


图 22.36　指定所用连接

单击“下一步”按钮，进入“设置参数”对话框，在该对话框的列表框中将显示参数名 custname，在右侧的下拉列表框中选择$F{custname}，如图 22.37 所示。

单击“下一步”按钮，进入到设置子报表路径的返回形式，这里选择以参数形式返回，如图 22.38 所示。

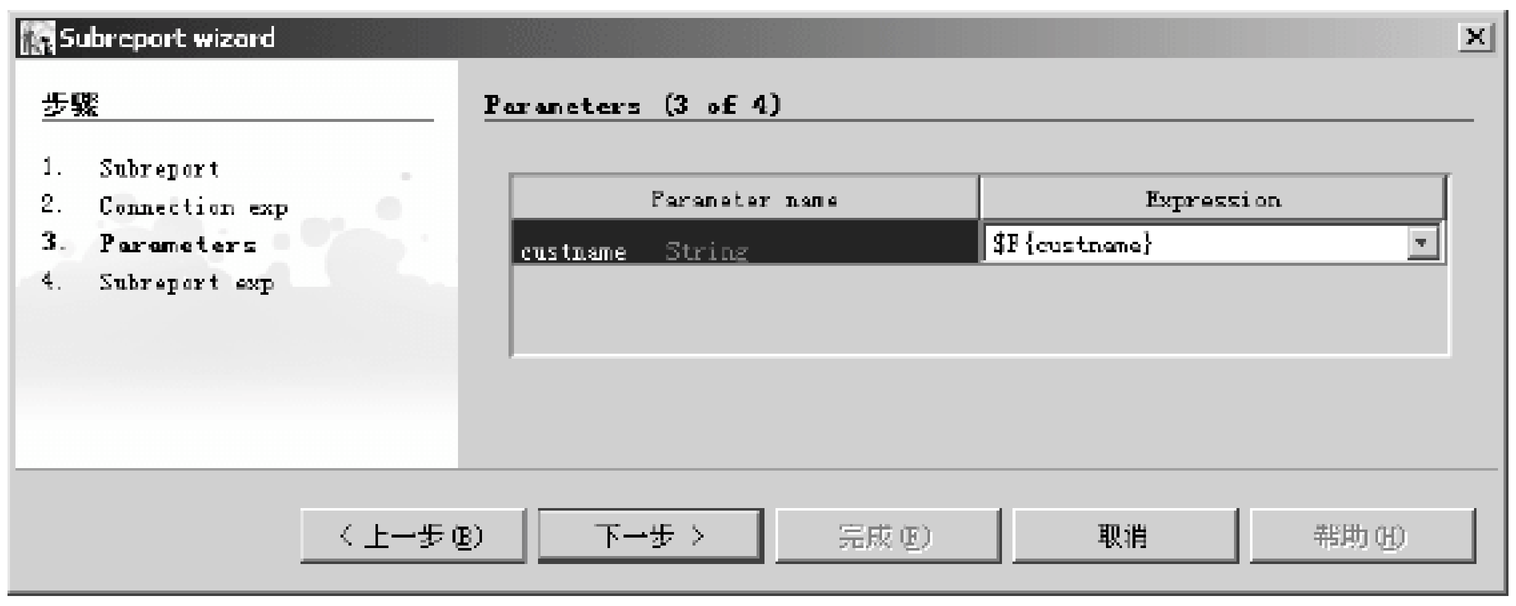


图 22.37　设置参数对话框

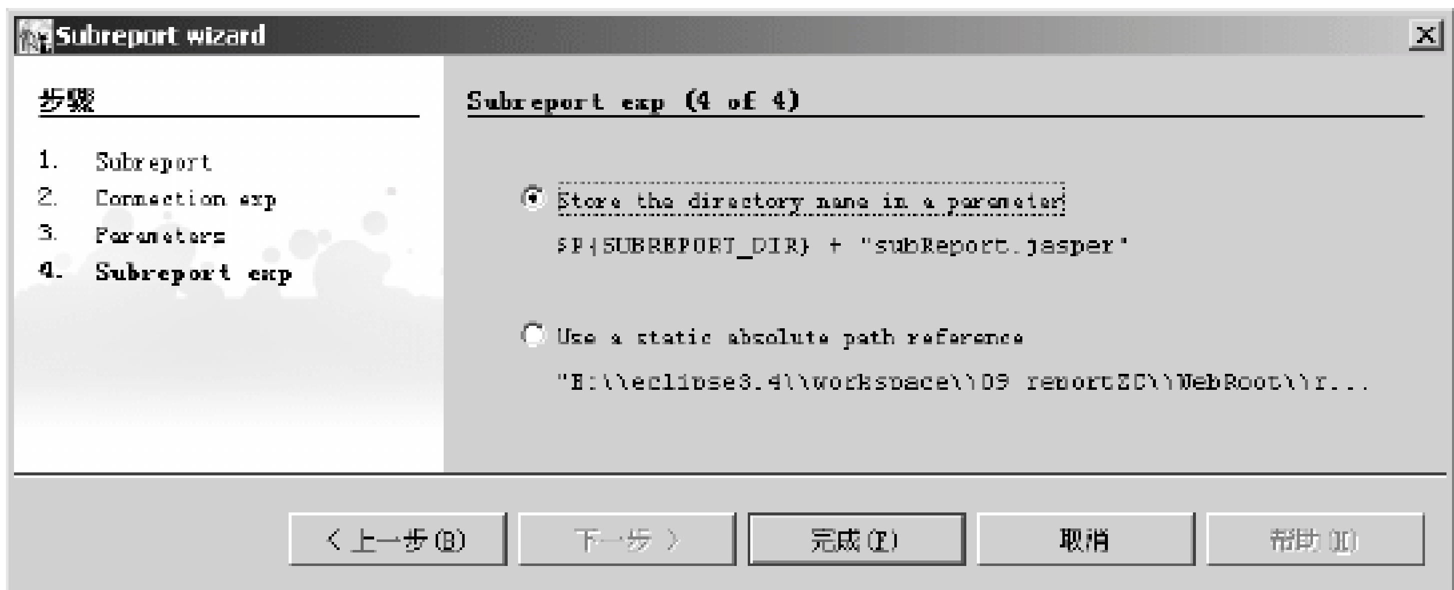


图 22.38　设置子报表路径的返回形式

单击“完成”按钮。这样在主报表的设计视图中将显示子报表，可以调整子报表的显示位置。

（13）至此，主报表设计完成，其最终的设计效果如图 22.39 所示。

（14）预览主从报表。选择主菜单中的 preview→PDF preview 命令，设置预览报表的格式为 PDF，单击 Preview 视图，预览报表，这时 iReport 会弹出提示对话框，要求输入相关参数的值，输入完成后，即可预览该报表。

（15）创建 Web 项目，将刚刚完成的主从报表复制到该项目的 reports 文件夹中，如图 22.40 所示。



图 22.39　子报表的最终设计效果

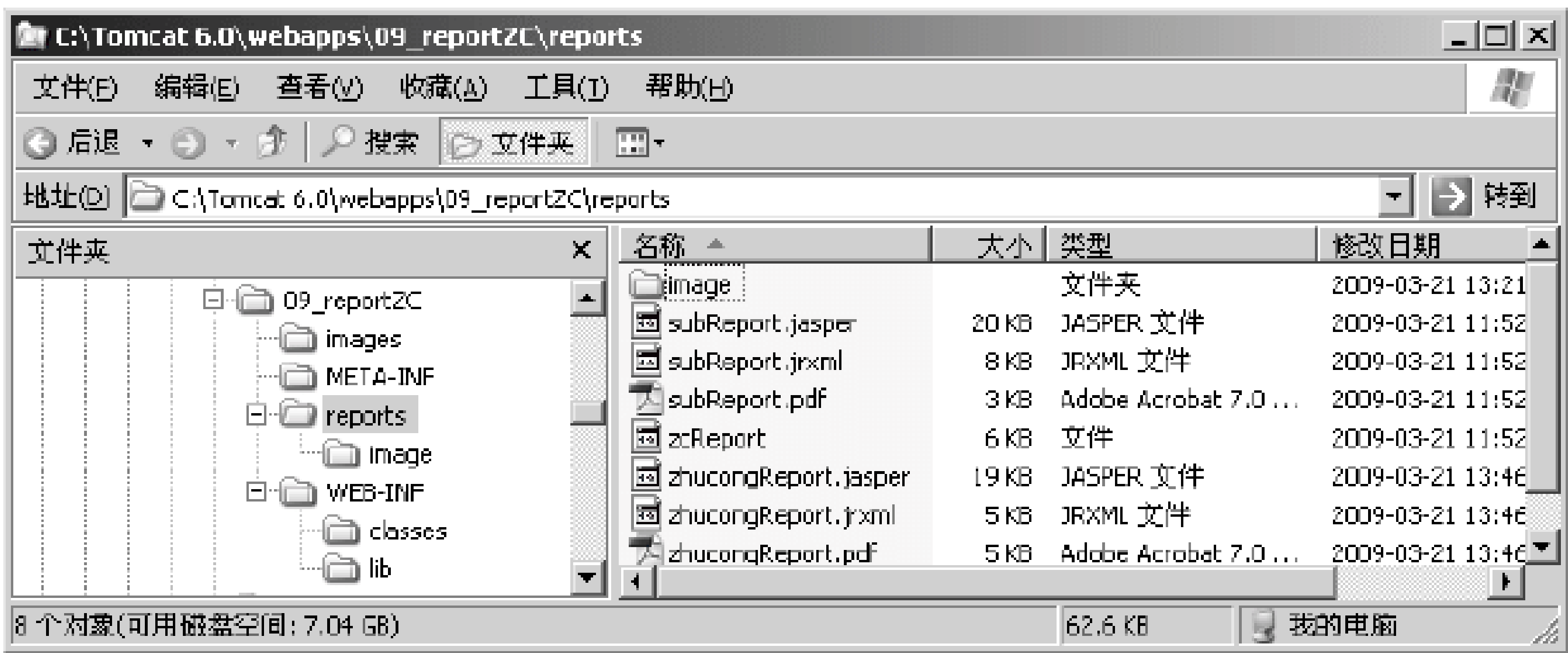


图 22.40　放置主从报表后的 Web 项目文件夹

（16）将该程序所需的 jar 包放置到 Web 项目的 WEB-INF\lib 文件夹下。编写 zc.jsp 文件，在该文件中调用主从报表，并指定相关参数。zc.jsp 文件的关键代码如下：

```
<%@ page import="net.sf.jasperreports.engine.*" %>
<%@ page import="net.sf.jasperreports.engine.data.*" %>
<%@ page import="net.sf.jasperreports.engine.export.*" %>
<%@ page import="net.sf.jasperreports.engine.util.*" %>
<%@ page import="java.util.*" %>
<%@ page import="java.sql.*" %>
<%@ page import="java.io.*" %>
<%
	JasperReport subReport=JasperCompileManager
 .compileReport(application.getRealPath("/reports/subReport.jrxml"));
	JasperReport fatherReport=JasperCompileManager
.compileReport(application.getRealPath("/reports/zhucongReport.jrxml"));
	File subreportFile = new File(application.getRealPath("/reports/subReport.jasper"));
	File zcReportFile = new File(application.getRealPath("/reports/zhucongReport.jasper"));
	Map<String,Object> parameters=new HashMap<String,Object>();
	String imgUrl=request.getRealPath("/reports/image/")+"/";
	imgUrl=imgUrl.replace("\\","/");
	parameters.put("url",imgUrl+"pdf_top.jpg");			//指定页眉图片的路径
	parameters.put("url_b",imgUrl+"pdf_bottom.jpg");		//指定页脚图片的路径
	//创建数据库连接
	Connection conn=null;
	try {
		Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver");
		conn=DriverManager.getConnection("jdbc:microsoft:sqlserver://localhost:1433;
DatabaseName=db_database09","sa","");
	}catch(Exception e) {
		e.printStackTrace();
	}
	parameters.put("Title", "SubReport");				//设置 Title 参数的值
	parameters.put("SubReportMap", conn);				//设置 SubReportMap 参数的值
	parameters.put("ChildReport", subReport);			//设置 ChildReport 参数的值
	parameters.put("SUBREPORT_DIR",request.getRealPath("/reports/")+"/");	//设置 SUBREPORT_DIR 参数的值
	//生成 PDF
	JasperRunManager run=new JasperRunManager();			//创建 JasperRunManager 对象的实例
	byte[] bytes = run.runReportToPdf(zcReportFile.getPath(), parameters, conn);
	response.setContentType("application/pdf");			//指定报表类型为 PDF
	response.setContentLength(bytes.length);			//设置内容长度
	ServletOutputStream ouputStream = response.getOutputStream();	//获取输出流对象
	ouputStream.write(bytes, 0, bytes.length);			//在输出流中写入内容
	ouputStream.flush();
	out.clear();
	out=pageContext.pushBody();
	ouputStream.close();						//关闭输出流对象
 %>
```

（17）编写 index.jsp 文件，在该文件中调用 zc.jsp 文件，显示生成的主从报表。index.jsp 文件的关键代码如下：

```
<body style="margin:0px;">
   <table width="599" border="0" align="center" cellpadding="0" cellspacing="0">
      <tr>
         <td><img src="images/main_top.jpg" width="599" height="191" border="0" usemap="#Map"></td>
      </tr>
   </table>
   <table width="599" border="0" align="center" cellpadding="0" cellspacing="0">
      <tr><td height="316" background="images/main_bottom.jpg"> </td></tr>
   </table>
   <iframe name="report" style="display:none"></iframe>
    <map name="Map"><area shape="rect" coords="138,168,202,188" href="zc.jsp" target="report"></map>
</body>
```

秘笈心法

本实例在应用 iReport 生成报表文件时，虽然实现过程的步骤非常多，但是这些步骤并不难，并且都是图形界面化的操作。最后在 JSP 页中，将生成的报表文件利用 JasperReport 组件处理即可。

实例 580 应用 iReport+JasperReport 生成分栏报表

光盘位置：光盘\MR\22\580

高级

实用指数：★★★

实例说明

分栏报表只是将数据信息以某个字段进行分类，然后以分栏的形式进行显示。分栏报表是比较常见的报表形式，它的最大优势就在于在打印报表时可以节约一定量的纸张。本实例将介绍如何在 JSP 中应用 iReport+JasperReport 生成分栏报表。运行程序，将显示如图 22.41 所示的页面，在该页面中，单击“工资管理”超链接，将按部门生成保存员工工资信息的 PDF 格式的分栏报表，并且通过 PDF 阅读器自动打开，如图 22.42 所示。



图 22.41　应用 iReport+JasperReport 生成分栏报表的主界面

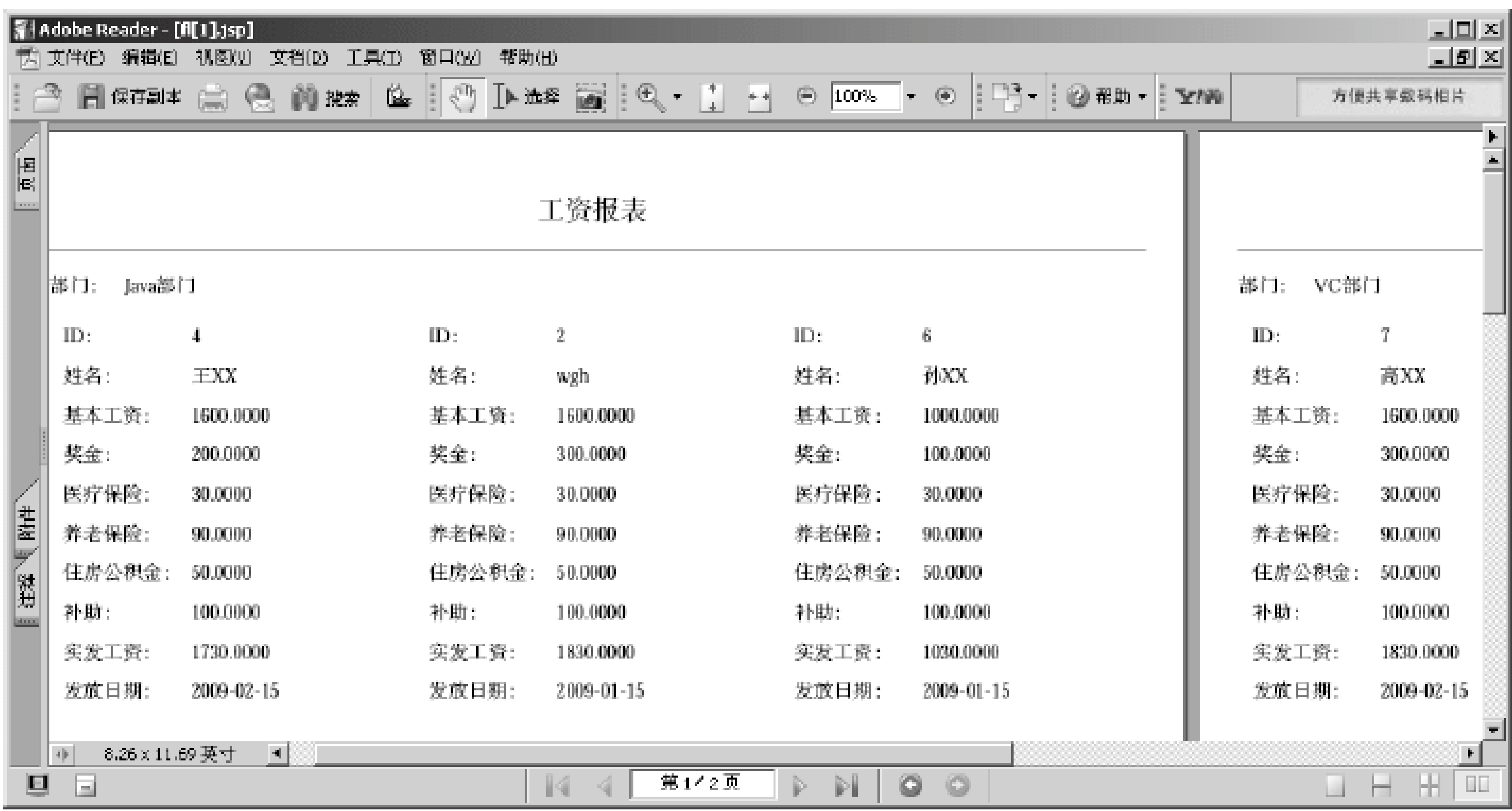


图 22.42　通过 PDF 阅读器打开生成的分栏报表

关键技术

本实例主要应用 iReport 与 JasperReport 组件实现。关于 iReport 和 JasperReport 组件的详细介绍可参见实例 579 的关键技术。

设计过程

（1）运行 iReport，选择主菜单中的“文件”→New→Empty report 命令，将打开 New 对话框，在该对话框的 Report name 文本框中输入报表名称 funlanReport，在 Location 文本框中输入该报表文件保存的路径，这里为

E:\09\fenlanReport，当然也可以通过单击 Browse 按钮选择文件保存的位置，单击“完成”按钮，完成报表文件的创建。

（2）新建报表完成后，需要为该报表配置相应的数据源，也就是指定该报表需要访问的数据库。

首先将所使用数据库的驱动包添加到 iReport 的 Classpath 中。选择主菜单中的“工具”→“选项”命令，在打开的“选项”对话框中，选择 iReport 节点下的 Classpath 选项卡，单击 Add JAR 按钮，在打开的添加 JAR 对话框中选择数据库驱动，这里为 SQL Server 2000 数据库的驱动包，单击“打开”按钮，即可将数据库驱动包添加到 Classpath 下，如图 22.43 所示。单击“确定”按钮，完成数据库驱动包的添加。



图 22.43 “选项”对话框的 Classpath 选项卡

然后就可以创建数据源。选择主菜单中的“工具”→Report Datasource 命令，将打开如图 22.44 所示的 Connections/Datasources 对话框。

在该对话框中单击 New 按钮，将打开 Datasource 对话框，在该对话框中选中 Database JDBC Connection 节点，单击 Next 按钮，将打开 Database JDBC connection 对话框，在该对话框的 Name 文本框中输入数据源名 db_database09，在 JDBC Driver 下拉列表框中选择数据库驱动名，这里为 MS SQL Server (2000) (com.microsoft.jdbc.sqlserver.SQLServerDriver)，修改 JDBC URL 文本框中生成的 JDBC URL 为 jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=db_database09，在 Server Address 文本框中输入服务器地址，这里为 localhost，在 Database 文本框中输入数据库名，这里为 db_database09，在 Username 文本框中输入用户名，这里为 sa，密码为空，选中 Save password 复选框，保存密码，如图 22.45 所示。

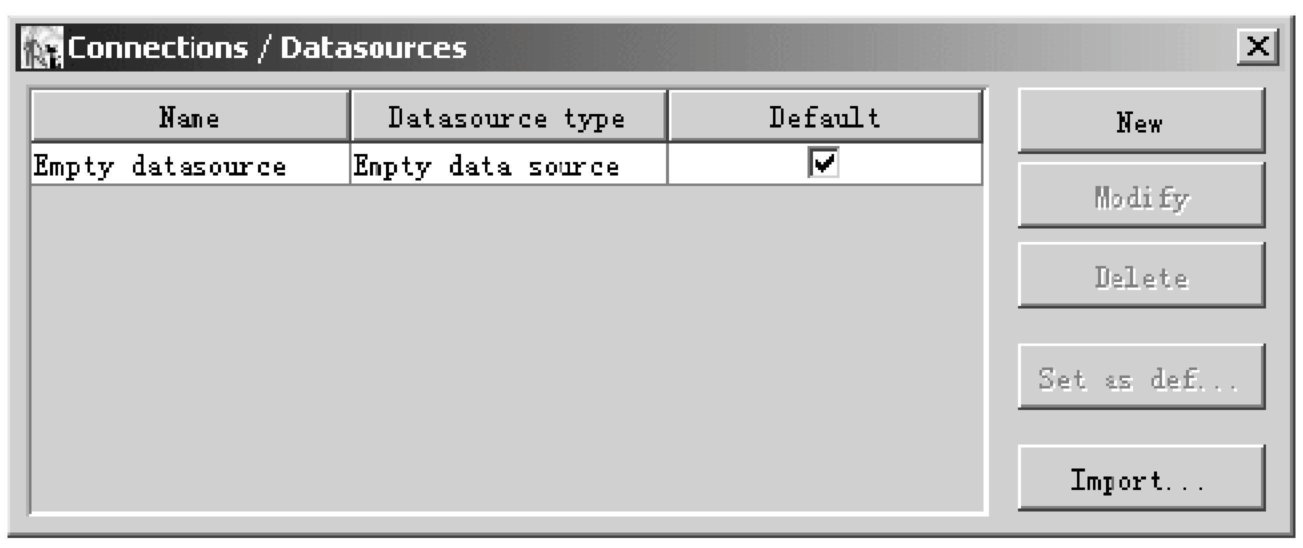


图 22.44 Connections/Datasources 对话框

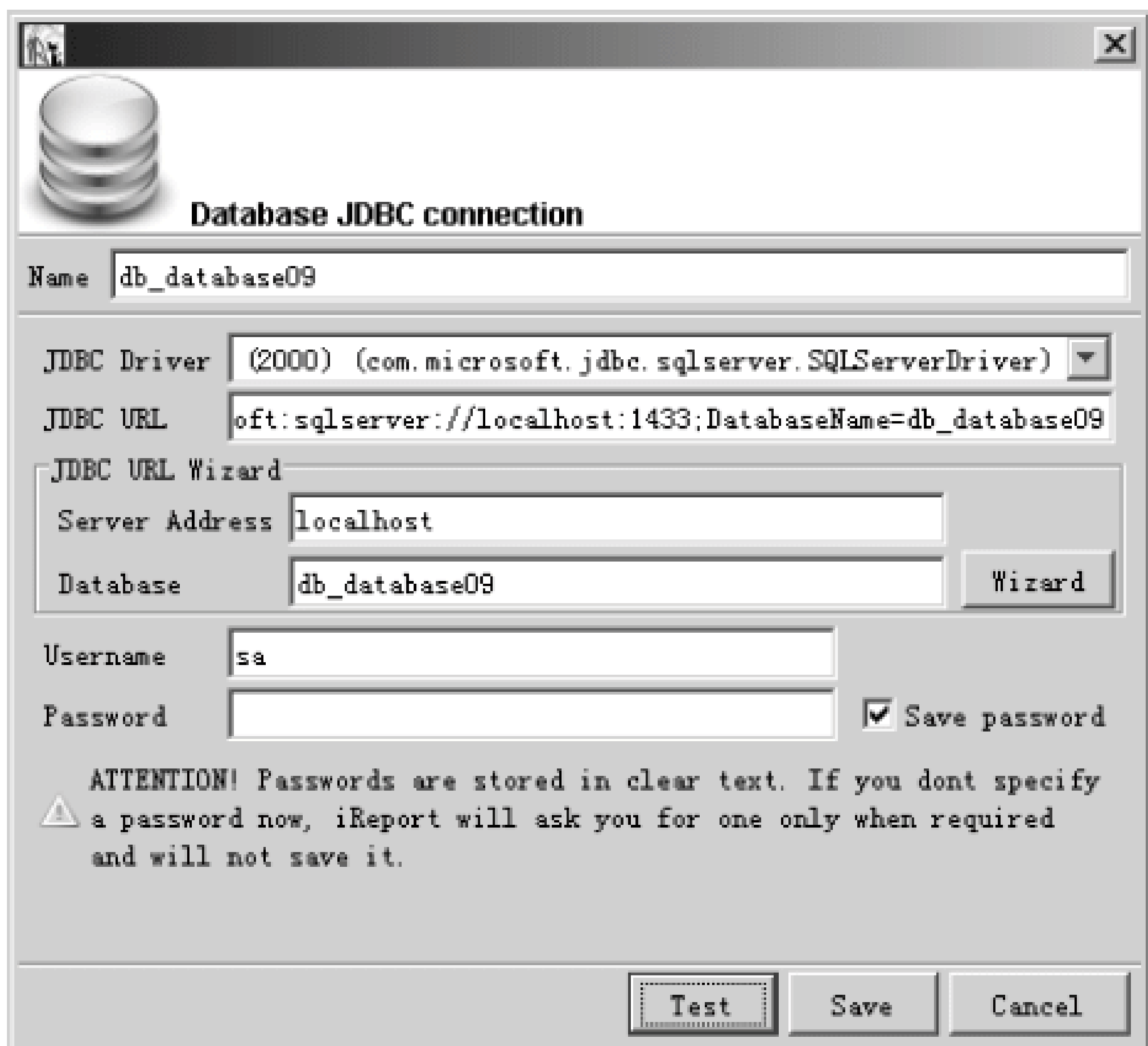


图 22.45 Database JDBC connection 对话框

单击 Test 按钮，测试数据源连接，如果成功，将显示 Connection test successful!对话框，否则将显示错误提示对话框，读者可以根据提示信息进行修改，直到提示连接成功。单击 Save 按钮，保存该数据源。这时该数据源将自动被选中。

（3）当数据源设置完成后，可以使用 SQL 语句指定报表中需要查询的数据。在 iReport 主界面左侧的 Report Inspector 面板中选中 report name 节点，并在该节点上单击鼠标右键，在弹出的快捷菜单中选择 Edit Query 命令，将打开 Report query 对话框，在该对话框的文本域中输入以下 SQL 语句：

```
select * from tb_personnelWage WHERE id <=$P{MaxOrderID} order by bm,ffTime desc
```

在该对话框下方的字段表格中将显示数据表中所有字段的相关信息，如图 22.46 所示。

设置完成报表查询的 SQL 语句，单击 OK 按钮即可。这时，在 Report Inspector 面板的 Fields 节点中将显示该查询结果中包括的全部字段。

（4）创建参数用于表示报表显示的行数，将其命名为 MaxOrderID，其具体属性设置如图 22.47 所示。

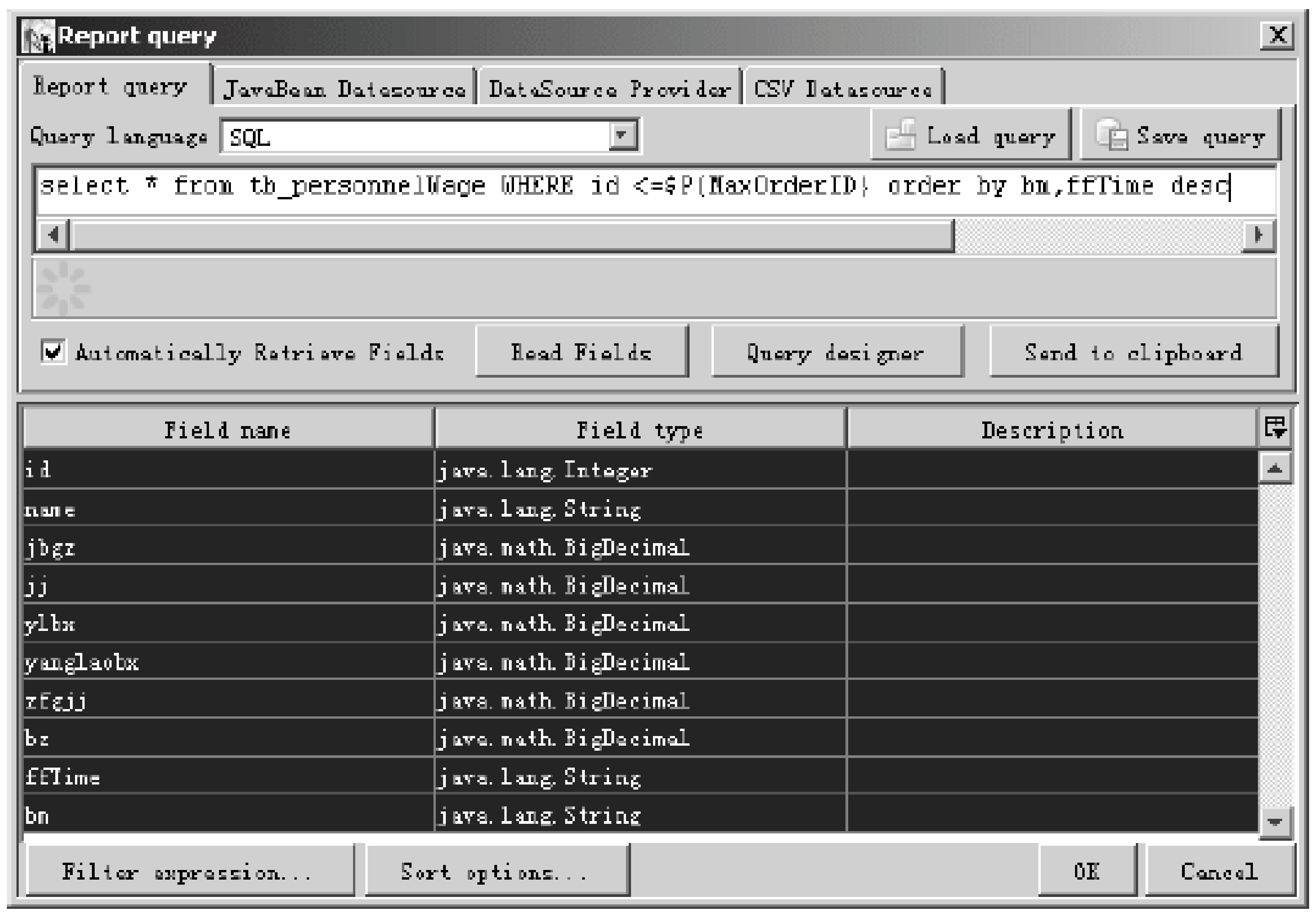


图 22.46　Report query 对话框



图 22.47　创建 MaxOrderID 参数

（5）按照步骤（2）介绍的方法再向 iReport 的 Classpath 中添加 jasperreports-3.1.4.jar 和 iTextAsian.jar 包，其中 iTextAsian.jar 用于解决生成报表不支持中文的问题。如果这两个包已经添加到 iReport 的 Classpath 中，此步骤可省略。

（6）创建参数用于表示报表标题，将其命名为 reportTitle，并设置其 Parameter Class 属性为 java.lang.String，其他采用默认。将该参数拖动到 Page Header 区域中，并设置 Page Header 区域的 Band Height 属性为 50。

（7）为报表设计分栏效果。在 Report Inspector 面板中，选中报表名称节点，这里将其设置为 fenlan，在属性面板中，将 Columns 区域的 Columns 属性设置为 3，表示该报表分为 3 栏。

（8）在 Report Inspector 面板中，选中报表名称节点，并单击鼠标右键，在弹出的快捷菜单中选择 Add Report Group 命令，将打开 New group wizard 对话框，在 Group name 文本框中输入分组名 bm；选中 Group by the following report object 单选按钮，在其下方的下拉列表框中选择要进行分组的字段，这里为部门字段 bm；其他采用默认，如图 22.48 所示。

（9）单击“下一步”按钮，将打开新建分组的详细对话框，在该页面采用默认设置，如图 22.49 所示。

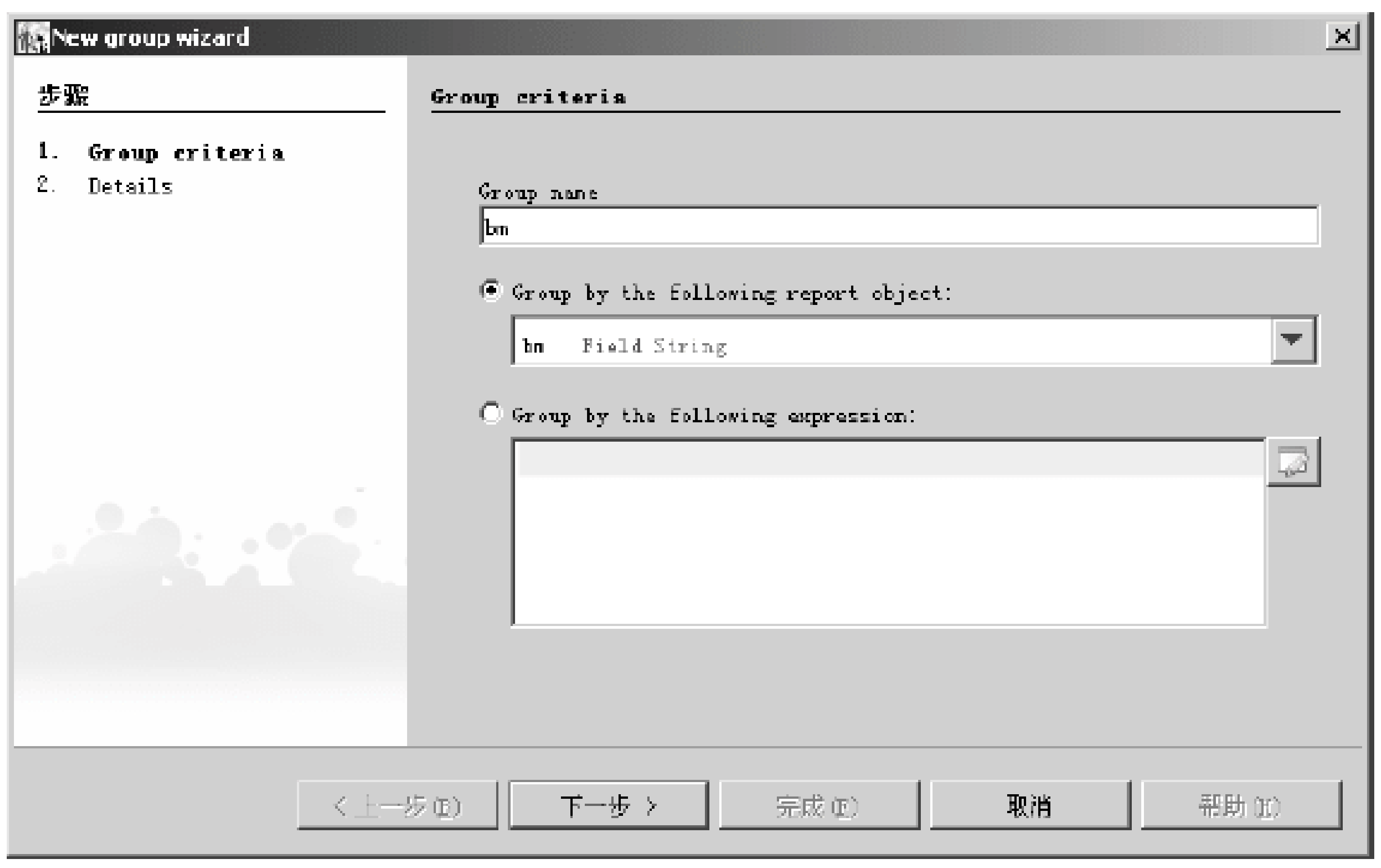


图 22.48　新建分组

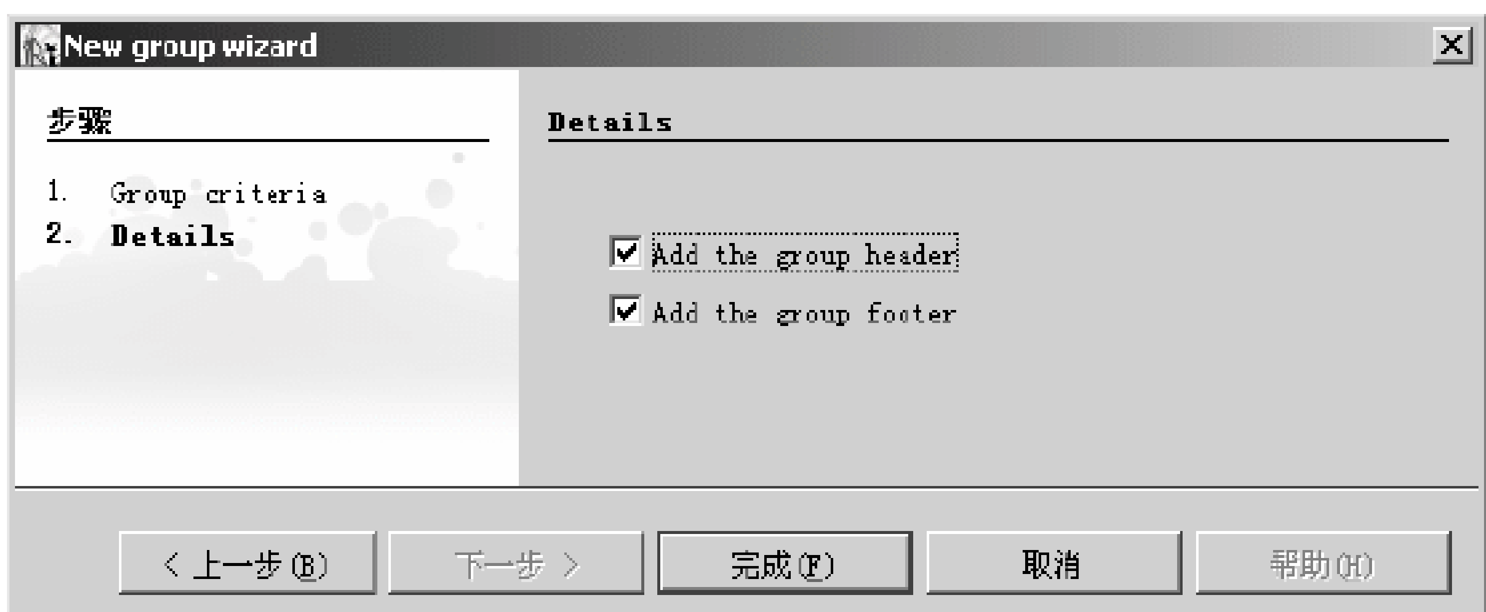


图 22.49　新建分组详细对话框

（10）单击“完成”按钮，完成新建分组，这时，在 Report Inspector 面板中将添加 bm Group Header 和 bm Group Footer 两个节点。此时，还需要将这两个节点的 Start on a new page 属性设置为选中状态。

（11）在 Report Inspector 面板中，将报表名称节点 fenlan 的 Pring order 属性设置为 Horizontal。

（12）在报表设计器的 bm Group Header 区域中添加一个 static Text 组件，并设置该组件的 Text 属性为“部门”；然后再添加一个 Text Field 组件，并设置该组件的 Text Field Expression 属性为表示部门的表达式$F{bm}。

（13）在报表设计器的 Detail 区域中，将字段名称与信息放置在合适的位置，设计完成的效果如图 22.50 所示。

（14）添加一个用于指定公司名称的参数 company，并将该参数的 Parameter Class 属性设置为 java.lang.String。

（15）在报表设计器的 Page Footer 区域中添加一个 Text Field，并设置该组件的 Text Field Expression 属性为参数表达式$P{company}。为了让该公司名称显示到页面的右侧，还需要设置该组件的 Horizontal Alignment 属性为 Right。至此，分栏报表设计完成。

（16）预览分栏报表。选择主菜单中的 preview→PDF preview 命令，设置预览报表的格式为 PDF，单击 Preview 视图，预览报表，这时 iReport 会弹出提示对话框，要求输入相关参数的值，输入完成后，即可预览该报表。

（17）创建 Web 项目，将刚刚完成的分栏报表复制到该项目的 reports 文件夹中，如图 22.51 所示。

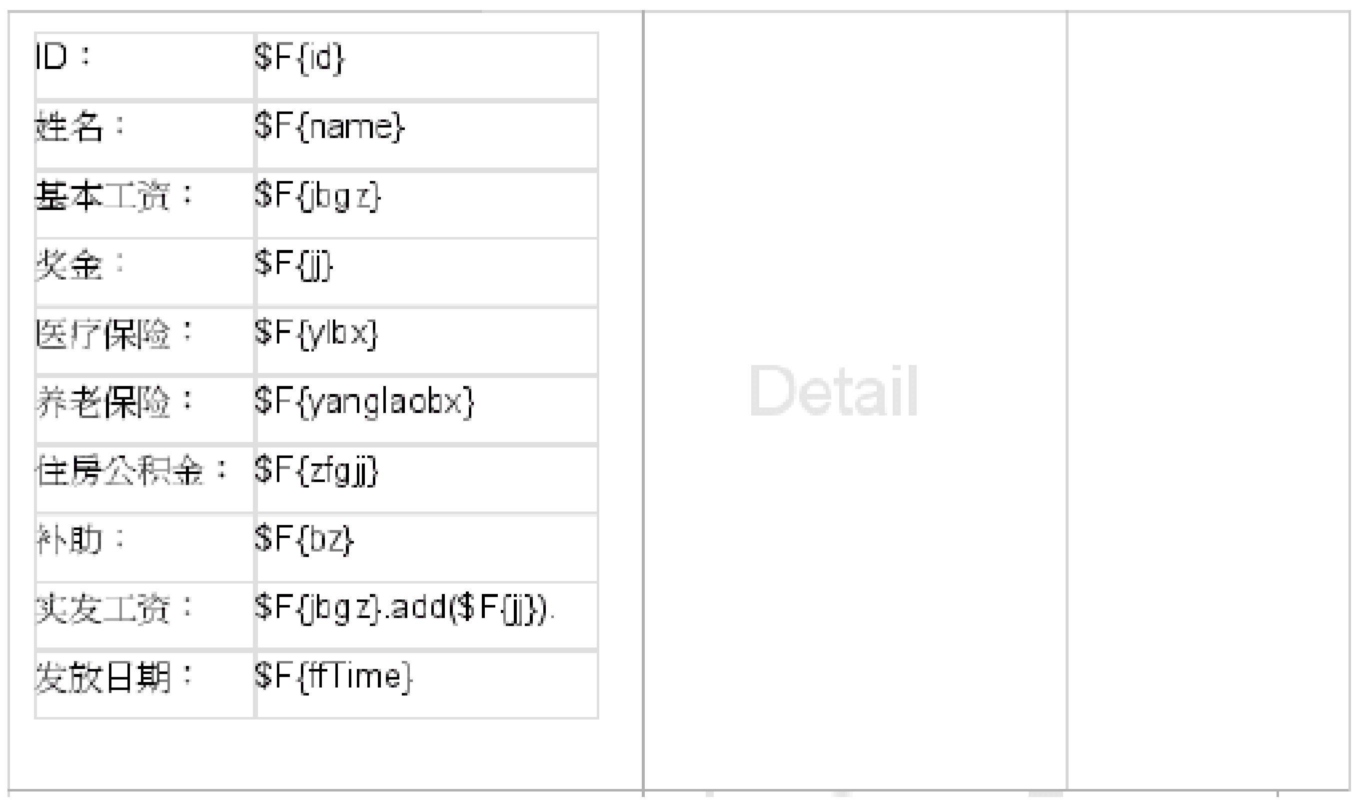


图 22.50　设计完成的 Detail 区域

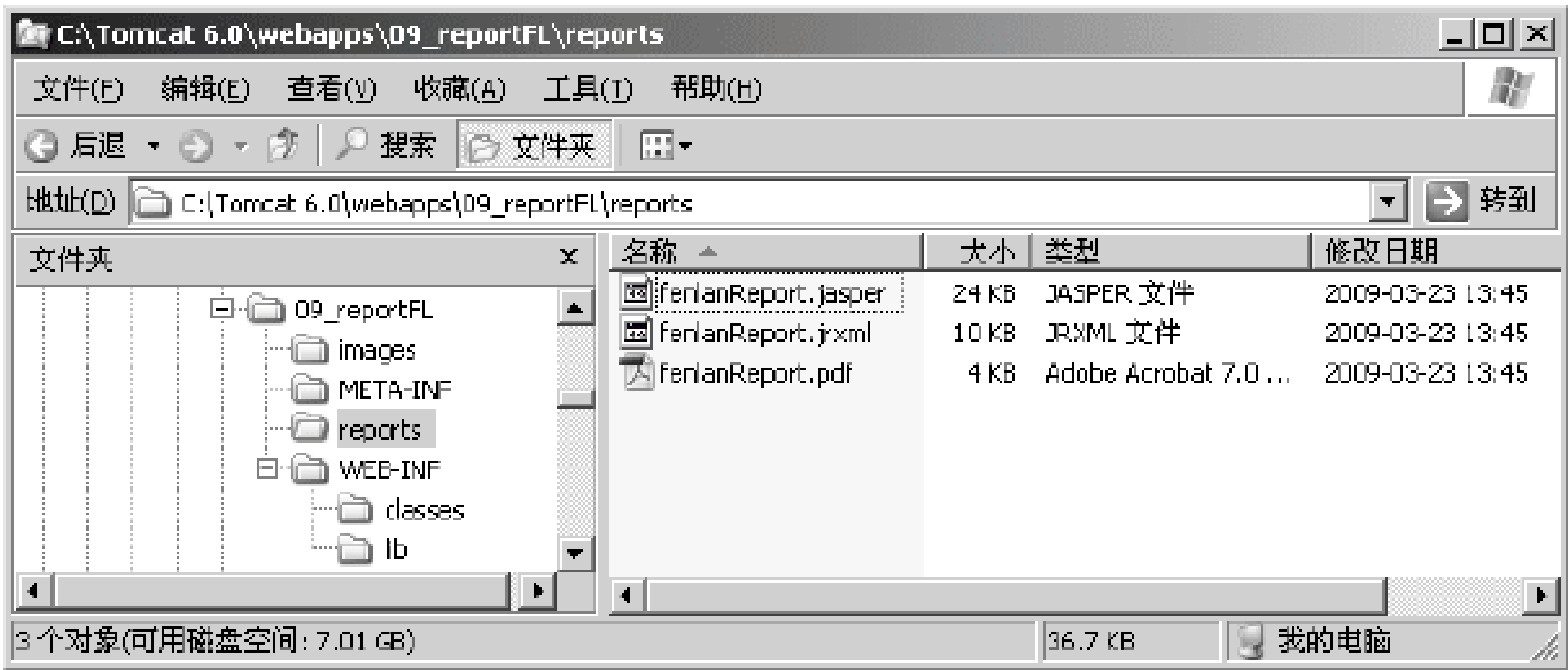


图 22.51　放置分栏报表后的 Web 项目文件夹

（18）将该程序所需的 jar 包放置到 Web 项目的 WEB-INF\lib 文件夹下。编写 fl.jsp 文件，在该文件中调用分栏报表，并指定相关参数。fl.jsp 文件的关键代码如下：

```
<%@ page import="net.sf.jasperreports.engine.*" %>
<%@ page import="net.sf.jasperreports.engine.data.*" %>
<%@ page import="net.sf.jasperreports.engine.export.*" %>
<%@ page import="net.sf.jasperreports.engine.util.*" %>
<%@ page import="java.util.*" %>
<%@ page import="java.sql.*" %>
<%@ page import="java.io.*" %>
<%
    JasperReport fatherReport=JasperCompileManager.compileReport(application.getRealPath("/reports/fenlanReport.jrxml"));
    File flReportFile = new File(application.getRealPath("/reports/fenlanReport.jasper"));
    Map<String,Object> parameters=new HashMap<String,Object>();
    //创建数据库连接
    Connection conn=null;
    try {
        Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver");
        conn=DriverManager.getConnection("jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=db_database22","sa","");

    }catch(Exception e) {
        e.printStackTrace();
    }
    int maxOrderId=10;
    parameters.put("reportTitle", "工资报表");                    //指定报表名称参数值
    parameters.put("company","吉林省明日科技有限公司");            //指定公司名称参数值
    parameters.put("MaxOrderID",maxOrderId);                      //指定最大记录数参数值
    //生成 PDF
    JasperRunManager run=new JasperRunManager();                  //创建 JasperRunManager 对象的实例
```

```
        byte[] bytes = run.runReportToPdf(flReportFile.getPath(), parameters, conn);
        response.setContentType("application/pdf");                        //指定报表类型为 PDF
        response.setContentLength(bytes.length);                           //设置内容长度
        ServletOutputStream ouputStream = response.getOutputStream();      //获取输出流对象
        ouputStream.write(bytes, 0, bytes.length);                         //在输出流中写入内容
        ouputStream.flush();
        out.clear();
        out=pageContext.pushBody();
        ouputStream.close();                                               //关闭输出流对象
%>
```

■ 秘笈心法

利用 iReport 工具生成报表文件之后，在 JSP 页中主要应用 JasperReport 组件，将报表文件中的.jasper 文件转换为字节数组，然后通过 Response 对象设置相应正文类型为 application/pdf，并输出表示报表的字节流。当访问该 JSP 页面时，会直接以 PDF 格式显示此报表信息。

第6篇

综合应用篇

⏭ 第23章　综合应用

第23章

综合应用

- 在线投票系统
- 用户注册
- 论坛
- 购物车
- 聊天室
- 万年历

23.1　在线投票系统

在线投票可以方便更多用户参与网上活动，同时真正调动广大网民参与的积极性。下面将通过几个实例介绍如何制作网上投票系统。

实例 581	禁止重复投票的在线投票系统 光盘位置：光盘\MR\23\581	初级 实用指数：★★★

实例说明

在开发 Web 程序时，经常需要实现一些投票功能。有些用户在使用时为了某种特殊的需求会恶意地投票，这样会严重影响投票结果的准确性。为了防止这种情况的发生，就要在程序中进行控制，让一个用户只能进行一次投票。运行本实例，如图 23.1 所示，没有投过票的用户第一次可以正常进行投票。当用户退回到投票页面想再次进行投票时便会出现提示，如图 23.2 所示。

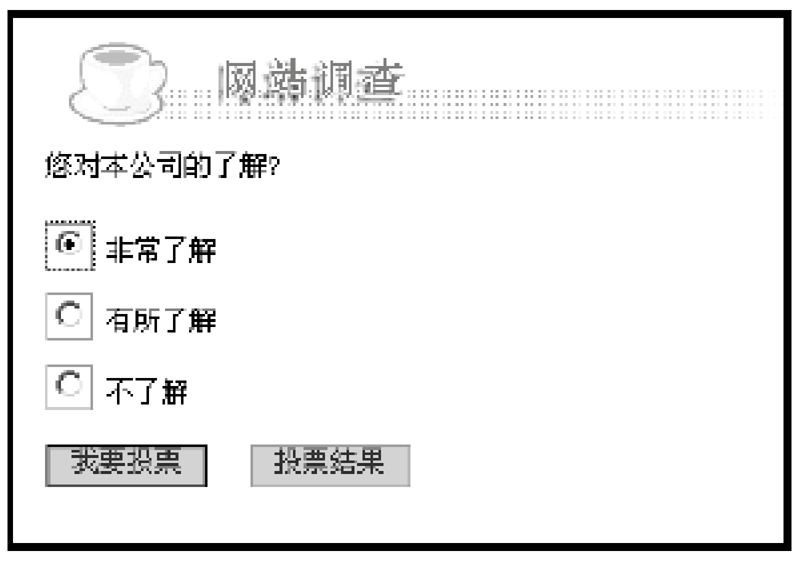


图 23.1　网上投票系统



图 23.2　禁止重复投票提示

关键技术

在实现本实例时主要用到了用户会话对象 session。该对象为 JSP 内置对象，Web 服务器会为每个用户创建一个单独的 session 对象，在本实例中把用户投票结果保存在该对象的属性中，在用户投票时通过该属性就可以判断出用户是否已经投过票了。

设计过程

（1）创建 JDBConnection.java 类文件，作用是取得对数据库的操作，读者可参考光盘中的源程序。

（2）创建 result.jsp 页面文件，用来处理用户的投票操作，关键代码如下：

```
<jsp:useBean id="connection" scope="request" class="com.dao.JDBConnection" />
<%
	String id = request.getParameter("id");	//获取用户选择的内容
	if (id == null) {	//如果用户没有选择
		response.sendRedirect("index.jsp");	//返回首页面
	} else {
		String sql = "update tb_voteOneTime set number=number+1 where id='" + id + "'";//生成 SQL 语句
		connection.executeUpdate(sql);	//调用执行 SQL 语句的方法
		connection.closeConnection();	//关闭数据库连接
		session.setAttribute("id",id);	//将用户选择的内容存入 session 对象
		response.sendRedirect("show.jsp");	//跳转到显示结果页面
	}
%>
```

（3）创建 show.jsp 页面文件，用于显示投票结果，关键代码如下：

```
<jsp:useBean id="connection" scope="request" class="com.dao.JDBConnection"/>
<table width="373" height="37" cellpadding="0" cellspacing="0">
  <tr>
    <td background="image/vote_info.gif"> </td>
  </tr>
```

```
</table>
<table width="373" height="57" cellpadding="0" cellspacing="0">
   <tr align="center">
      <td width="122" height="33"><div align="left">序号 </div></td>
      <td width="138"><div align="center">投票百分比</div></td>
      <td width="121"><div align="center">投票人数</div></td>
   </tr>
   <%
String selectNumber="select sum(number) as number from tb_voteOneTime";          //生成计算总投票人数的 SQL 语句
ResultSet ret=connection.executeQuery(selectNumber);                             //执行 SQL 语句，获取结果集
String strNumber="";                                                             //声明用于保存投票人数的变量
while(ret.next()){
              strNumber=ret.getString("number");                                 //获取总投票人数
        }
ResultSet rs=connection.executeQuery("select * from tb_voteOneTime order by id");   //查询数据表
while(rs.next()){
        float singleNumber=Float.parseFloat(rs.getString("number"));              //当前项的票数
        float allNumber=Float.valueOf(strNumber);                                 //总投票人数
        float result=singleNumber/allNumber*100;                                  //计算出百分比
%>
   <tr align="center">
      <td height="20"><div align="left"><%=rs.getString("id")%>.<%=rs.getString("name")%></div></td>
      <td><%=result%>%</td>
      <td><%=rs.getString("number")%></td>
   </tr>
   <%
        }
connection.closeConnection();
%>
</table>
<table width="373" height="24" cellpadding="0" cellspacing="0">
   <tr>
      <td>共有[<%=strNumber%>]人参加投票</td>
   </tr>
</table>
```

秘笈心法

本实例主要利用 session 对象实现，如果同一时刻有多个客户与服务器在进行会话，那么在 Servlet 容器中会存在多个 HttpSession 对象。针对这一特点，可以利用 session 来实现在线投票系统，还可以实现购物车。

实例 582	每个 IP 一个月只能投票一次的投票系统 光盘位置：光盘\MR\23\582	初级 实用指数：★★★

实例说明

很多网站的开发者都想了解浏览者对该网站的一些看法和意见，这个功能可以通过用户投票来实现。本实例是一个对编程语言使用率的调查。运行本实例，如图 23.3 所示，选择所使用的一种编程语言后，单击“我要投票”按钮进行网络投票。单击“投票结果”按钮即可查看网上投票情况。当客户端已经投过票后，再次单击“我要投票”按钮时将弹出提示信息，如图 23.4 所示。

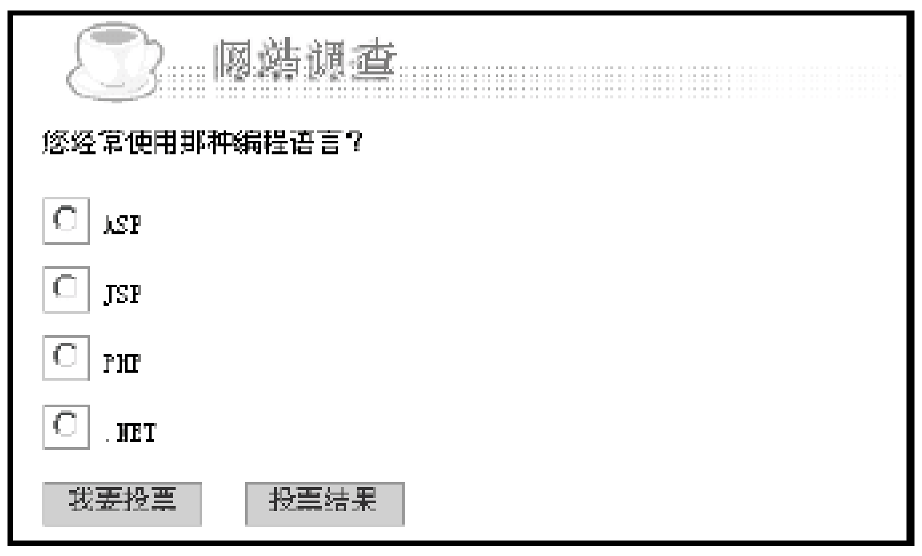


图 23.3　选择常用的编程语言



图 23.4　提示信息

关键技术

本实例主要应用 Cookie 实现一个 IP 地址每月只能投一次票的功能。通过 request.getRemoteHost()方法取得客户端的 IP 地址后，将其存放在 Cookie 对象中，并且通过 cookie.setMaxAge (60*60*24*30)方法设置存储的时间。

设计过程

（1）创建 JDBConnection.java 类文件，作用是取得对数据库的操作，读者可参考光盘中的源程序。

（2）实现投票处理的关键代码如下：

```
<%
    String IP = request.getRemoteHost();          //获得当前客户端的 IP 地址
    Cookie[] cookies = request.getCookies();      //取得客户端 Cookie
    boolean flag = true;
    for (int i = 0; i < cookies.length; i++) {
      if (IP.equals(cookies[i].getValue())) {
        flag = false;                             //如果当前 IP 与存放在 Cookie 对象中的 IP 一致，则将对象 flag 设置为 false
      }
    }
    if(flag){                                     //如果不一致，新建一个 Cookie
    String id = request.getParameter("id");       //获取用户选择的内容编号
    String sqlVote = "update tb_VoteIP set number=number+1 where id='" + id + "'";        //生成 SQL 语句
    connection.executeUpdate(sqlVote);            //执行 SQL 语句
    connection.closeConnection();                 //关闭数据库连接
    Cookie cookie = new Cookie("IP", IP);         //声明 Cookie 对象
    cookie.setMaxAge(60*60*24*30);                //设置保存 Cookie 的时间
    response.addCookie(cookie);                   //添加 Cookie 对象
    response.sendRedirect("show.jsp");            //转到显示结果页面
    }else{                                        //如果一致，提示用户已经投过票了
out.print("<script language='javascript'>alert('同一台电脑使用同一 IP 地址每月只能投一次票');window.location.href='index.jsp';</script>");
    }
  %>
```

秘笈心法

本实例主要利用 Cookie 技术实现。在实现本实例时，首先获取到当前客户端的 IP 地址，然后再与 Cookie 中保存的 IP 进行比较，如果相同，则提示已经投过票了，否则将当前客户端的 IP 存放在 Cookie 中。

23.2　用 户 注 册

如果要提高网站的安全性，防止非法用户进入网站，可在用户进入网站前先进行注册，只有注册成功的用户才可以进入网站。下面将通过几个实例介绍如何设计用户注册。

实例 583	带检测用户名的用户注册 光盘位置：光盘\MR\23\583	初级 实用指数：★★★

实例说明

在开发用户注册模块时，用户名是用来标识用户身份的，所以在数据库中要确保用户注册的用户名是唯一的。用户可以在输入完用户名后，对自己输入的名称进行检测。运行本实例，在“用户名”文本框中输入用户名，单击“检测用户”超链接对输入的用户名进行检测，判断该用户名是否已经存在。如果已经存在，则提示用户重新输入用户名，否则用户可以进行其他相关信息的注册，程序运行界面如图 23.5 所示。

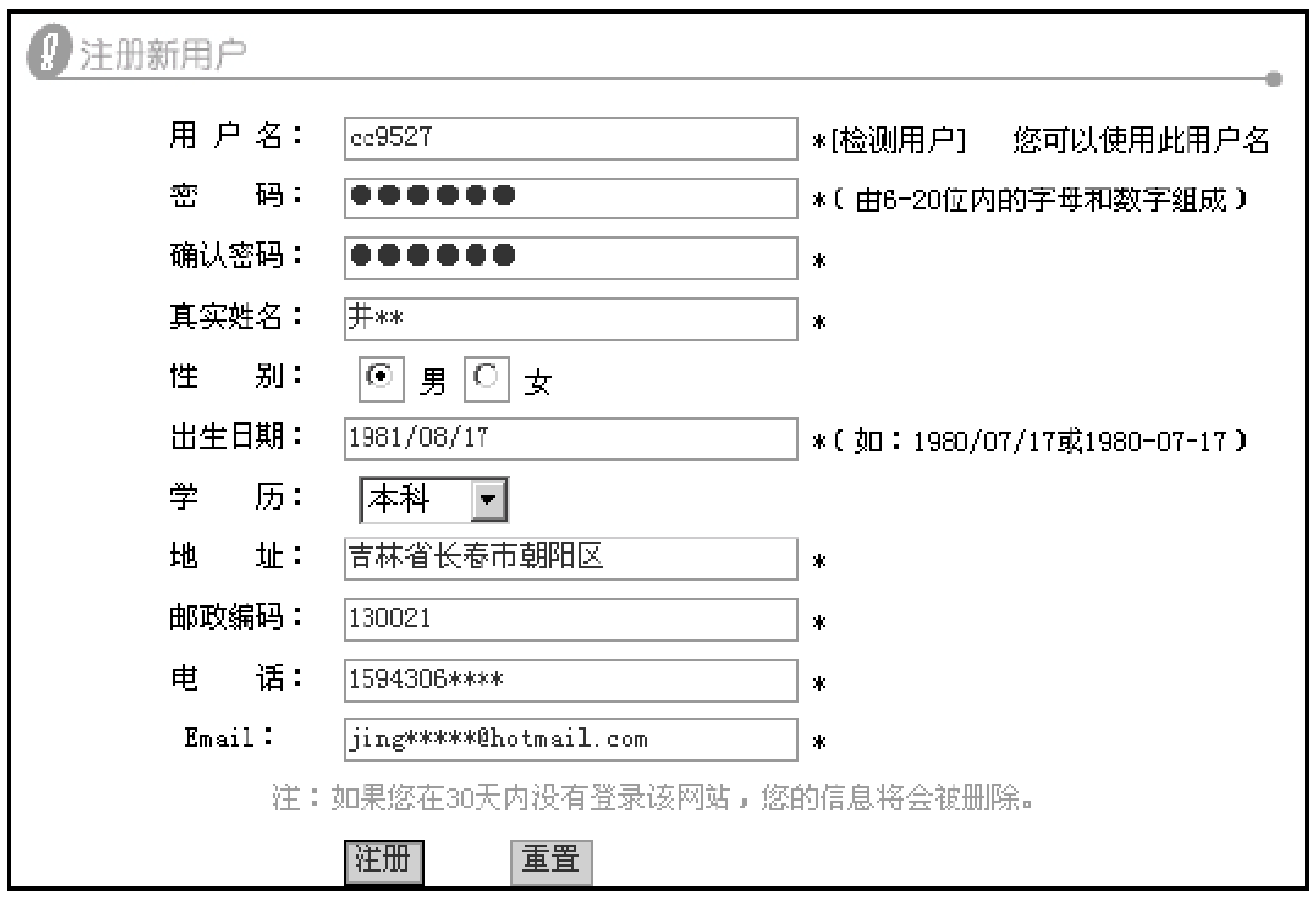


图 23.5　带检测用户名的用户注册

关键技术

实现本实例的设计思路是，在“用户名”文本框中输入要注册的用户名，单击“检测用户”超链接，链接到 check.jsp 页面中检测用户名是否存在，并且把检测结果传递到首页面中，在“检测用户”超链接后面显示出来。

设计过程

（1）单击“检测用户”超连链将调用 JavaScript 脚本，关键代码如下：

```
<script language="javascript" type="">
function openwin(){
if (form1.account.value==""){
alert("请输入用户名!");
return false;
}
var str="check.jsp?account="+form1.account.value;
window.location.href=str;
}
</script>
  *[<a href="#" onClick="openwin()">检测用户</a>]
```

（2）判断注册的用户名是否已经存在的关键代码如下：

```
<jsp:useBean id="connection" scope="request" class="com.JDBConnection"/>
<%
request.setCharacterEncoding("GBK");
String account=request.getParameter("account");
String sql="select * from tb_checkUpName where account='"+account+"'";
ResultSet rs=connection.executeQuery(sql);
boolean flag=true;
String result="您可以使用此用户名";
try{
while(rs.next()){
flag=false;
result="该用户名已经被使用";
}
}catch(Exception e){}
connection.closeConnection();
if(flag){
%>
<script language="javascript" type="">window.location.href='index.jsp?account=<%=account%>&result=<%=result%>'</script>
<%}else{%>
<script language="javascript" type="">window.location.href='index.jsp?result=<%=result%>'</script>
<%}%>
```

（3）将新注册的用户信息添加到指定的数据表中的关键代码如下：

```
   <jsp:useBean id="connection" scope="request" class="com.JDBConnection"/>
  <%
     request.setCharacterEncoding("GBK");                       //设置字符集
     String account = request.getParameter("account");           //获取表单中的信息
     String password = request.getParameter("password1");
```

```
        String name = request.getParameter("name");
        String sex = request.getParameter("sex");
        String bron = request.getParameter("bron");
        String knowledge = request.getParameter("knowledge");
        String address = request.getParameter("address");
        String post = request.getParameter("post");
        String tel = request.getParameter("tel");
        String email = request.getParameter("email");
        String sql = "insert into tb_checkUpName values ('" + account + "','" + password + "','" + name + "','" + sex + "','" + bron + "','" + knowledge + "','" + 
address + "','" + post + "','" + tel + "','" + email + "')";                    //生成 SQL 语句
        String success = "";
        if (connection.executeUpdate(sql)) {                                   //执行 SQL 语句，并判断是否执行成功
          success = "用户注册成功";
        }else {
          success = "用户注册失败";
        }%>
<script language="javascript" type="">window.location.href='index.jsp?success=<%=success%>'</script>
```

秘笈心法

需要注意的是，在 tb_checkUpName 表中，把 account 字段设置为主键，当新用户注册时，主键值不能有重复的，输入相同的用户名时，对数据表添加的操作会失败。

实例 584　分步用户注册

光盘位置：光盘\MR\23\584

初级

实用指数：★★★

实例说明

用户注册的形式有很多种，本实例实现了用户的分步注册功能。用户的分步注册可以让用户更清楚地知道用户具有哪些权限或哪些特殊的能力。运行程序，单击“我接受”按钮，表示在注册成为用户时已经确认该服务条款，并进入下一页面；如果单击“我拒绝”按钮，将不能再进行用户注册。进入“选择用户名”页面时，输入用户的相关信息后单击“提交表单”按钮，将进入“填写个人资料”页面，在此页面中将添加个人的详细信息。程序运行结果如图 23.6～图 23.8 所示。

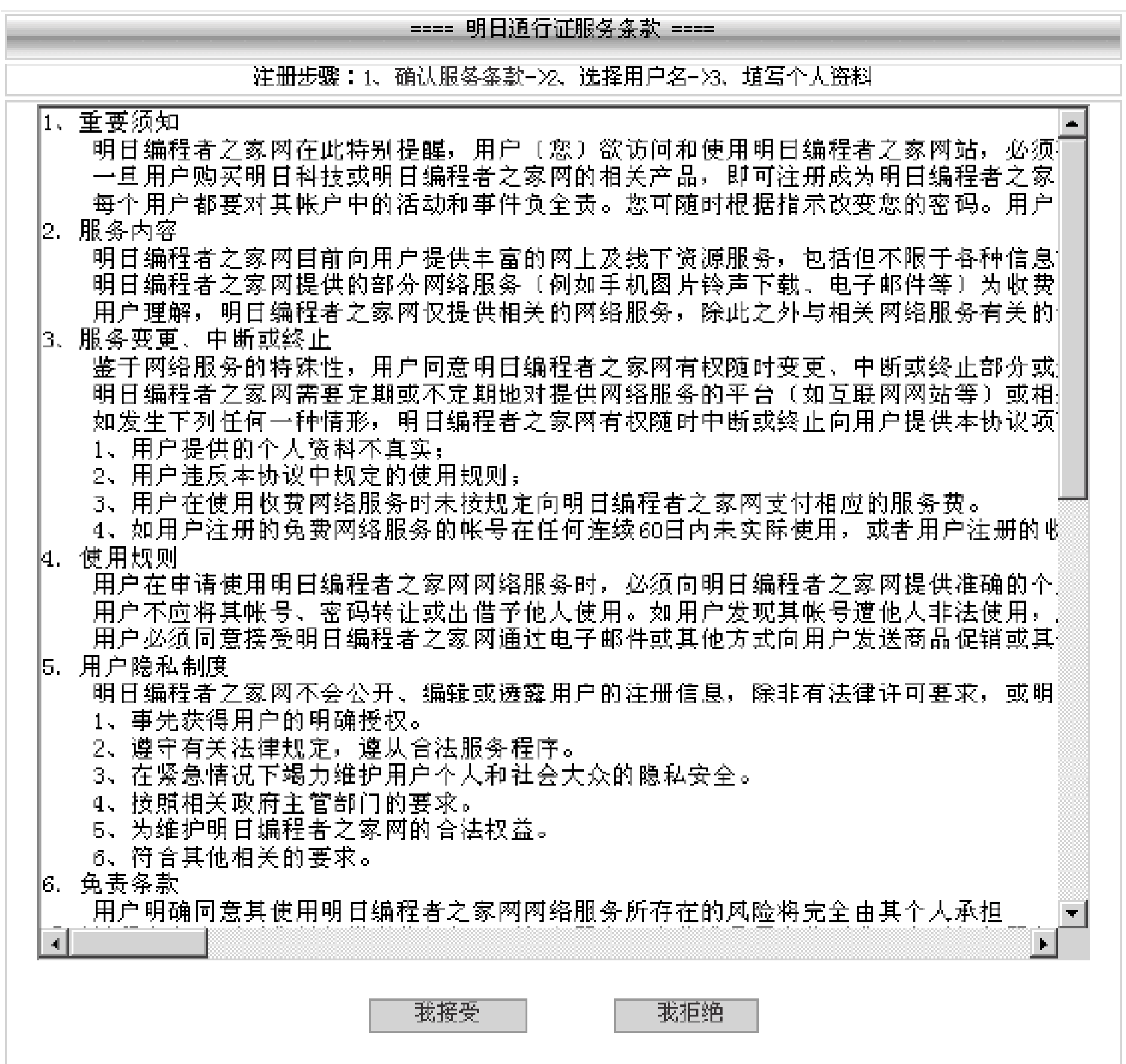


图 23.6　分步用户注册 1



图 23.7　分步用户注册 2



图 23.8　分步用户注册 3

关键技术

本实例主要应用 INSERT INTO 语句将用户注册的相关信息添加到数据表中。本实例的实现过程是：首先进入服务条款项目；当用户接受服务条款之后，将进行用户名、登录密码、确认密码和密码问题等相关信息的输入，输入完成后提交表单；最后进入输入个人信息的页面，输入结束后再次提交表单，至此完成用户的分步注册。

设计过程

（1）首先确认用户是否接受服务条款，单击“我接受”按钮将进入用户注册的相关页面。

（2）通过 JavaScript 脚本对输入的相关信息格式进行限定，必须输入正确的格式，其关键代码如下：

```
<script language="javascript" type="text/javascript">
function Mycheck(){
	if(form1.username.value==""){
		alert("请输入正确的用户名!! ");
		form1.username.focus();
		return;
	}
	if(form1.pass1.value==""){
		alert("请您正确地输入登录密码(仅可用英文、数字!!)");
		form1.pass1.focus();
		return;
	}
	if(form1.pass2.value==""){
		alert("请输入登录密码确认！ ");
		form1.pass2.focus();
		return;
	}
	if(form1.pass1.value!=form1.pass2.value){
		alert("您两次输入的密码不一致，请重新输入！ ");
		form1.pass1.focus();
		return;
	}
	if(form1.question.value==""){
		alert("请输入提示问题，当您忘记密码时可根据该问题提示密码！ ");
		form1.question.focus();
		return;
	}
	if(form1.answer.value==""){
		alert("请输入问题答案！ ");
		form1.answer.focus();
		return;
	}
	form1.submit();
}
</script>
```

（3）实现分步用户注册 2 页面的添加用户的操作，以添加数据是否成功为条件判断用户名是否存在的关键代码如下：

```
<jsp:useBean id="connection" scope="request" class="com.JDBConnection"/>
<%
request.setCharacterEncoding("GBK");                    //设置字符集
String username=request.getParameter("username");       //获取页表单中的信息
String password=request.getParameter("pass1");
String question=request.getParameter("question");
String answer=request.getParameter("answer");
String sql="insert into tb_rPartEnroll (account,password,question,answer) values ('"+username+"','"+password+"','"+question+"', '"+answer+"')";
if(connection.executeUpdate(sql)){                      //执行 SQL 语句，并判断是否执行成功
%>
<script language="javascript" type="text/javascript">window.location.href='resNext.jsp?username=<%=username%>'</script>
<%}else{%>
<script language="javascript" type="text/javascript">window.location.href='res.jsp?username=<%=username%>'</script>
<%}%>
```

（4）实现对分步用户注册 3 页面的用户做修改操作的关键代码如下：

```
   <jsp:useBean id="connection" scope="request" class="com.JDBConnection"/>
 <%
   request.setCharacterEncoding("GBK");                 //设置字符集
   String username = request.getParameter("username");  //获取表单信息
   String realName = request.getParameter("realName");
   String sex = request.getParameter("sex");
   String bron = request.getParameter("bron");
   String number = request.getParameter("number");
   String email = request.getParameter("email");
   String sql ="update tb_rPartEnroll set realName='" + realName + "',sex='" + sex + "',bron='" + bron + "',number='" + number + "', email='" + email + "'
where account='" + username + "'";                      //生成 SQL 语句
  String result = "";
   if (connection.executeUpdate(sql)) {                 //执行 SQL 语句
     result = "注册成功!!! ";
   }else {
     result = "注册失败!!! ";
   }
 %>
<script language="javascript" type="text/javascript">window.location.href='index.jsp?result=<%=result%>'</script>
```

秘笈心法

分步用户注册其实并不难，主要就是将每一步的表单信息都更新到数据库表中即可。例如，本实例首先将步骤（2）中的表单信息添加到数据库的 tb_rPartEnroll 表中，然后在步骤（3）中应用 update 语句更新 tb_rPartEnroll 表的其他字段数据。

实例 585	通过 E-mail 激活的用户注册 光盘位置：光盘\MR\23\585	初级 实用指数：★★★

实例说明

为了防止别有用心的人恶意注册，可以使用通过 E-mail 激活注册用户的方法来激活新注册的用户。运行本实例，单击“新用户注册”链接后如图 23.9 所示。输入注册信息后单击“提交信息”按钮，激活注册用户的邮件将会被发送到用户填写的邮箱中，如图 23.10 所示。通过访问邮件中的地址就可以实现激活用户的操作。

关键技术

要运行本实例，首先应该构建 JavaMail 的开发环境，在 JDK 中配置 JavaMail 的相关类和包。

在构建 JavaMail 开发环境中，需要 mail.jar 和 activation.jar 两个文件。这两个文件的获得可以通过 SUN 公司的官方网站下载。本实例在 WEB-INF\lib 文件夹中存放的就是这两个文件。



图 23.9　用户注册页面

请点击下面的连接激活用户，如果不能点击请手动复制到地址栏中执行
http://192.168.1.42:8080/453/Activation?id=1&name=111

图 23.10　激活用户邮件内容

JavaMail 对 SMTP、POP3、IMAP 提供支持，封装了电子邮件功能中的邮件对象、发送、身份验证、接收等功能。

在发送各种类型的邮件时，主要应用到下面几个类。

（1）Session 类。用户要想发送邮件首先需要创建 Session 类的对象，利用该对象创建邮件对象、指定邮件服务器认证的客户端属性。其类层次结构为 javax.mail.Session。

（2）InternetAddress 类。邮件发送的地址类，其类层次结构为 javax.mail.internet.InternetAddress，它继承自抽象类 javax.mail.Address。

（3）MimeMessage 类。邮件消息类，其类层次结构为 javax.mail.internet.MimeMessage，继承自抽象类 javax.mail.Message。

（4）Transport 类。邮件发送类，其类层次结构为 javax.mail.Transport。

（5）Authenticator 类。授权者类，JavaMail 通过使用 Authenticator 类以用户名、密码的方式访问那些受到保护的资源，在这里“资源”就是指邮件服务器。其类层次结构为 javax.mail. Authenticator。

（6）Store 类。用来从邮件服务器上接收邮件，其类层次结构为 javax.mail.Store。

（7）Folder 类。邮件文件夹类，其类层次结构为 javax.mail.Folder。

设计过程

（1）创建 DBConnection 类，用于获取数据库连接。创建 ActivUserDto 类，用于封装数据信息。

（2）创建 ActivUserDto 类，用于对数据表进行操作。在该类中编写 insert()方法，用于注册新用户，并将新用户的编号返回，关键代码如下：

```
//用户注册，发送激活用户邮件
public int insert(ActivUserDto dto) {
    Connection con = null;
    PreparedStatement ps = null;
    ResultSet rs = null;
    int id = 0;
    try {
        con = DBConnection.getConnection();
        ps = con.prepareStatement("INSERT INTO tb_activuser values (default,?,?,?,'0')", PreparedStatement.RETURN_GENERATED_KEYS);
        ps.setString(1, dto.getName());
        ps.setString(2, dto.getPwd());
        ps.setString(3, dto.getMail());
      if (ps.executeUpdate() == 1) {                    //如果插入成功
          rs = ps.getGeneratedKeys();                   //得到自动生成的主键编号
          while (rs.next()) {
              id = rs.getInt(1);
          }
```

```
    }
    ……//省略部分代码
```

另外在该类中编写 activUser()方法，用于按照（id,name）将数据表中的用户账号激活，关键代码如下：

```
public void activUser(Integer id, String name) {
    Connection con = null;
    PreparedStatement ps = null;
    try {
        con = DBConnection.getConnection();
        ps = con.prepareStatement("UPDATE tb_activuser SET state=1 WHERE id=? and name=?");
        ps.setInt(1, id);
        ps.setString(2, name);
        ps.execute();
        ps.close();
        con.close();
    ……//省略部分代码
```

（3）创建 SendMail 类，在该类中编写 sendMail()方法，发送用于从资源文件中读取要发送激活邮件所用到的邮箱配置信息，并向用户发送激活邮件，关键代码如下：

```
public void sendMail(String mail,String url) {
    InputStream is = this.getClass().getResourceAsStream("/mailInfo.properties");
    Properties prop = new Properties();
    try {
        prop.load(is);                                          //加载资源文件
    } catch (IOException e1) {
        e1.printStackTrace();
    }
    String msgText = "请点击下面的连接激活用户，如果不能点击请手动复制到地址栏中执行\n" + url;
    String smtpHost = prop.get("smtpHost").toString();          //SMTP 服务器名
    String from = prop.get("mailName").toString();              //发信人地址
    String pwd = prop.get("pwd").toString();                    //密码
    String to = mail;                                           //收信人地址
    Properties props = new Properties();                        //创建 Properties 对象
    props.put("mail.smtp.host", smtpHost);                      //创建邮件服务器
    Session session = Session.getDefaultInstance(props, null);  //取得默认的 Session
    MimeMessage message = new MimeMessage(session);             //创建一条信息，并定义发信人地址和收信人地址
    try {
        message.setFrom(new InternetAddress(from));
        InternetAddress[] address = {new InternetAddress(to)};
        message.setRecipients(Message.RecipientType.TO, address);
        message.setSubject("激活注册用户");                      //设定主题
        message.setSentDate(new Date());                        //设定发送时间
        message.setText(msgText);                               //把前面定义的 msgText 中的文字设定为邮件正文的内容
        message.saveChanges();                                  //保存发送信息
        Transport transport = session.getTransport("smtp");     //协议
        transport.connect(smtpHost, from, pwd);                 //发信人地址、用户名、密码
        transport.sendMessage(message, message.getAllRecipients());
        transport.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

（4）创建 RegServlet 类。RegServlet 类是向数据表中插入新用户信息，同时发送激活邮件的 servlet。该类首先实现了 doGet()与 doPost()方法，它们分别执行 HTTP 中 get 类型的请求与 post 类型的请求，在本实例中这两种类型的请求都通过调用 processRequest()方法来实现业务逻辑，processRequest()方法将用户输入的注册信息插入数据表中，并为用户发送激活邮件，关键代码如下：

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=GBK");
    ActivUserDto dto = new ActivUserDto();
    dto.setName(request.getParameter("name"));
    dto.setPwd(request.getParameter("pwd1"));
    dto.setMail(request.getParameter("mail"));
    int id = new ActivUserDao().insert(dto);
    String url = "http://";
    url+=request.getLocalAddr()+":";
```

```
        url+=request.getLocalPort();
        url+=request.getContextPath();
        url+="/Activation";
        url += "?id="+id+"&name="+dto.getName();
        new SendMail().sendMail(dto.getMail(), url);

        PrintWriter out = response.getWriter();
        out.print("<h3 align='center'>用户注册完成，激活账号邮件已经发出，请您登录您的邮箱按照信中地址激活您的账号</h3>");
        out.print("<br><center><a href='index.jsp'>返回首页</a></center>");
        out.close();
}
```

（5）创建 Activation 类，Activation 类是用来处理用户激活请求的 servlet。该类首先实现了 doGet()与 doPost()方法，它们分别执行 HTTP 中 get 类型的请求与 post 类型的请求，在本实例中这两种类型的请求都通过调用 processRequest()方法来实现业务逻辑，processRequest()方法通过获取 URL 中的参数激活用户，关键代码如下：

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        response.setContentType("text/html;charset=GBK");
        Integer id = Integer.valueOf(request.getParameter("id"));
        String name = request.getParameter("name");
        new ActivUserDao().activUser(id, name);
        PrintWriter out = response.getWriter();
        out.print("<br><br>");
        out.print("<center>用户"+name+"已被激活</center>");
        out.print("<br><center><a href='index.jsp'>进入登录页面</a></center>");
        out.close();
}
```

（6）在 web.xml 文件中配置 RegServlet 类和 Activation 类，首先使用<servlet-name>与<servlet-class>标签配置 servlet 的名称与所在的包名类名，然后再通过<url-pattern>标签配置 servlet 的映射路径，关键代码如下：

```
<servlet>
        <servlet-name>RegServlet</servlet-name>
        <servlet-class>com.jwy.servlet.RegServlet</servlet-class>
</servlet>
<servlet>
        <servlet-name>Activation</servlet-name>
        <servlet-class>com.jwy.servlet.Activation</servlet-class>
</servlet>
<servlet-mapping>
        <servlet-name>RegServlet</servlet-name>
        <url-pattern>/RegServlet</url-pattern>
</servlet-mapping>
<servlet-mapping>
        <servlet-name>Activation</servlet-name>
        <url-pattern>/Activation</url-pattern>
</servlet-mapping>
```

（7）编写 index.jsp 页面，用于登录或进入新用户注册页面，关键代码如下：

```
<form name="f1" method="post" action="login.jsp" onsubmit="return check()">
        <table border="1" align="center">
                <tr>
                        <td>用户名：</td>
                        <td><input type="text" name="name"></td>
                </tr>
                <tr>
                        <td>密    码：</td>
                        <td><input type="password" name="pwd"></td>
                </tr>
                <tr>
                        <td colspan="2" align="center">
                                <input type="submit" value="登录">      
                                <a href="reg.jsp">注册新用户</a>
                        </td>
                </tr>
        </table>
</form>
```

（8）编写 reg.jsp 页面，用于注册新用户，关键代码如下：

```
<form name="f1" method="post" action="RegServlet" onsubmit="return check()">
    <table align="center" border="1">
        <tr>
            <td>用户名：</td>
            <td><input type="text" name="name"></td>
        </tr>
        <tr>
            <td>密码：</td>
            <td><input type="password" name="pwd1"></td>
        </tr>
        <tr>
            <td>确认密码：</td>
            <td><input type="password" name="pwd2"></td>
        </tr>
        <tr>
            <td>电子邮箱：</td>
            <td><input type="text" name="mail"></td>
        </tr>
        <tr>
            <td colspan="2" align="center">
                <input type="submit" value="注册" />      
                <input type="reset" value="重置"/>
            </td>
        </tr>
    </table>
</form>
```

（9）在 mailInfo.properties 资源文件中配置用户自己发送激活邮件所用的邮箱的主机地址、邮箱账号及邮箱密码，关键代码如下：

```
smtpHost=192.168.*.**
mailName=jing***@163.com
pwd=***
```

秘笈心法

本实例在通过邮件激活用户时，给用户填写的邮箱地址中发送了一封含有 URL 地址的电子邮件，将用户 ID 与用户名作为参数保存在 URL 地址中，当用户通过此 URL 地址访问指定页面时，该页面通过获取 URL 中的参数来实现激活用户的功能。

23.3　论　　坛

随着网络的普及，论坛的内容越来越丰富，论坛也由此深受广大网民的喜爱。因此，商业网站对论坛也重视起来，纷纷在自己的网站上开辟论坛，提供与网民交流的平台，同时在线技术支持和在线服务也在论坛中开展起来。下面通过 5 个典型实例对论坛中的相关技术进行详细介绍。

实例 586	查看帖子信息 光盘位置：光盘\MR\23\586	初级 实用指数：★★★

实例说明

帖子信息就是供别人浏览查看并分享自己的想法和经验的信息。进入关于“明日科技出版的[JSP]类图书专区”论坛首页，单击讨论主题超链接，将进入到查看主题信息的页面，该页面主要以列表形式浏览帖子的主题和帖子的全部内容。运行结果如图 23.11 所示。

关键技术

IE 地址栏中的 id 值为主题信息表（tb_forumSend）中的字段 id 值，例如，程序发布后，在 IE 地址栏中输

入“http://localhost:8080/forum/showContent.jsp?id=1”，将值 1 传递到 showContent.jsp 页面，然后通过 Request 对象获取地址栏中的 id 值，最后应用 SQL 语句查询 id 值所对应的主题信息。执行 SQL 语句的关键代码如下：

```
String sqlSend="select * from tb_forumSend where id='"+request.getParameter("id")+"'";
```



图 23.11　查看帖子信息

设计过程

（1）创建 JDBConnection.java 类文件，主要作用是取得对数据库的操作，代码读者可参考光盘中的源程序。

（2）检索数据库中的数据，根据用户选择的主题 id 显示主题信息，关键代码如下：

```
<jsp:useBean id="connection" scope="request" class="com.JDBConnection"/>            //应用 JavaBean
<%
String right=(String)session.getAttribute("right");                                 //获取保存在 session 对象中的数据
if(right==null){                                                                    //如果该对象不存在
right="";
}
String sqlSend="select * from tb_forumSend where id='"+request.getParameter("id")+"'"; //根据编号获取指定留言信息
ResultSet rsSend=connection.executeQuery(sqlSend);                                  //执行 SQL 语句
String title="";
String account="MrFriend";
String content="";
String ip="";
String creatime="";
String sex="";
try{
while(rsSend.next()){                                                               //循环遍历查询结果集
title=rsSend.getString("title");                                                    //获取主题信息
content=rsSend.getString("content");                                                //获取内容信息
account=rsSend.getString("account");                                                //获取留言者信息
ip=rsSend.getString("ip");                                                          //获取编号信息
creatime=rsSend.getString("creatime");                                              //获取时间信息
String sqlUser="select * from tb_forumUser where account='"+account+"'";
ResultSet rsUser=connection.executeQuery(sqlUser);
while(rsUser.next()){
sex=rsUser.getString("sex");
}
%>
<%}}catch(Exception e){}%>
```

（3）检索数据库中的数据，根据用户选择的主题 id 查询回复信息，关键代码如下：

```
<%
String reSign="";
String reTitle="";
String reContent="";
String reAccount="";
String recreatime="";
String reSex="";
String sqlReyle="select tb_forumBack.*,tb_forumUser.sex from tb_forumBack inner join tb_forumUser on tb_forumBack.id= tb_forumUser.id where
tb_forumUser.id='"+request.getParameter("id")+"'";                                  //查询回复信息 SQL 语句
ResultSet rsReyle=connection.executeQuery(sqlReyle);                                //执行 SQL，获取查询结果集
try{
```

```
while(rsReyle.next()){                          //循环遍历查询结果集
reSign=rsReyle.getString("sign");               //获取 tb_forumBack 表中的 sign 属性
reTitle=rsReyle.getString("title");             //获取论坛回复主题
reContent=rsReyle.getString("content");         //获取论坛回复内容
reAccount=rsReyle.getString("account");         //获取论坛回复人
recreatime=rsReyle.getString("creatime");       //获取回复时间
reSex=rsReyle.getString("sex");                 //获取回复人性别
%>
<%}}catch(Exception e){}%>
```

秘笈心法

实现本实例，主要是根据用户选择相应的帖子主题，然后根据帖子主题的 id 查询数据库中主题表的信息，并查询对应的回复信息。

实例 587	发表主题信息 光盘位置：光盘\MR\23\587	初级 实用指数：★★★

实例说明

发表主题信息是为了在论坛中互相讨论话题而产生的，发表主题信息就是表达自己的看法，与他人进行讨论，因此发表主题是论坛首要的功能。用户在查看主题信息页面中单击“发表主题”超链接，可进入“发表主题”页面发表主题信息，运行结果如图 23.12 所示。

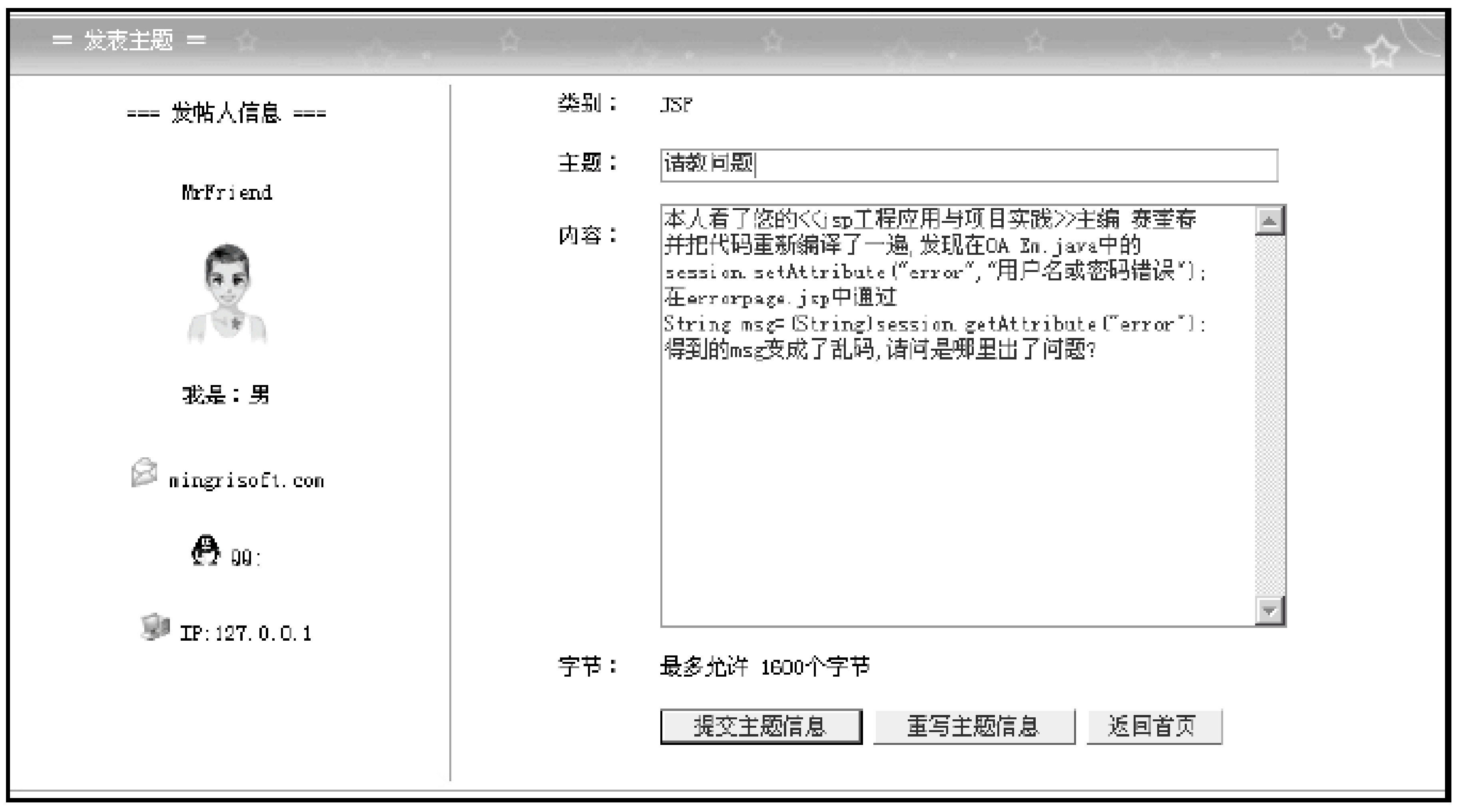


图 23.12　发表主题

关键技术

在开发专业论坛网站的过程中需要考虑到发布主题的人以及发布时间。在实际操作中，这些不需要用户输入，系统会根据实际情况自动填写。

设计过程

（1）创建 ShowTime.java 类文件，主要作用是通过 showTodayTime()方法取得系统的时间，程序代码如下：

```
package com;
import java.text.DateFormat;
import java.util.Date;                                  //导入需要的类包
public class ShowTime {
    public String showTodayTime(){                      //创建获取系统时间方法
    Date date=new Date();
    return DateFormat.getDateInstance().format(date);   //系统时间格式化返回
    }
}
```

（2）利用 JSP 内置对象 Request 接收发表的主题信息，然后将用户发表的信息保存在主题表中，程序代码如下：

```
<jsp:useBean id="connection" scope="request" class="com.JDBConnection"/>
<jsp:useBean id="showTime" scope="request" class="com.ShowTime"/>                    //应用 JavaBean
<%
request.setCharacterEncoding("gbk");                                                 //设置请求信息编码
String url=request.getParameter("url");                                              //获取请求参数
String account=request.getParameter("account");
String ip=request.getParameter("ip");
String sort=request.getParameter("sort");
String title=request.getParameter("title");
String content=request.getParameter("content");
String sql="insert into tb_forumSend values ('"+sort+"','"+title+"','"+content+"','"+account+"','"+ip+"','"+showTime.showTodayTime()+"')"; connection.
executeUpdate(sql)                                                       //执行 SQL 语句
connection.closeConnection();                                                        //关闭连接
%>
```

秘笈心法

发表主题信息，也就是向主题信息表中插入一条数据，用户只需要输入主题和内容，其他字段信息会自动添加，如 IP 地址和发表时间等。

实例 588	回复主题信息 光盘位置：光盘\MR\23\588	初级 实用指数：★★★

实例说明

当用户浏览主题时，可能会对该主题有自己的看法，此时用户可以在查看主题信息页面上单击“回复”超链接，进入到“回复主题”页面来发表自己的意见和想法。运行结果如图 23.13 所示。

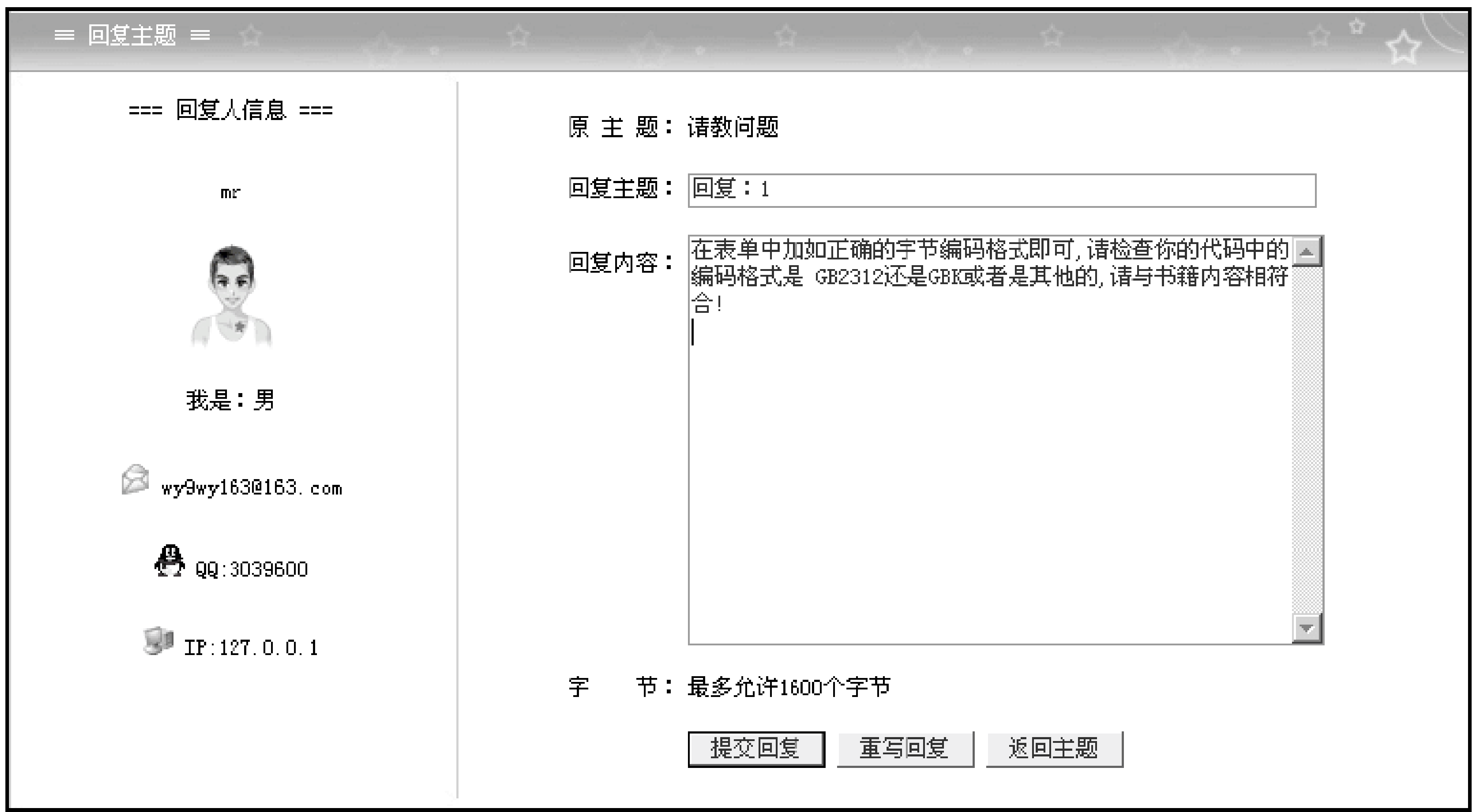


图 23.13　回复主题信息

关键技术

将参数 id 值作为回复信息表（tb_forumBack）中的 id（字段 id）值应用到程序中，例如，在 IE 地址栏中输入“http://localhost:8080/forum/backInformation.jsp?id=5”，将值 5 传递到 backInformation.jsp 页面后，通过 Request 对象获取地址栏中的 id 值。最后应用 INSERT 语句将回复信息添加到数据库中。回复主题信息的 SQL 语句的代码如下：

```
String sqlReyle="select tb_forumBack.*,tb_forumUser.sex from tb_forumBack inner join tb_forumUser on tb_forumBack.account= tb_forumUser.account
where tb_forumBack.id='"+request.getParameter("id")+"'";
```

设计过程

通过 Request 对象获取回复主题信息的内容，利用 INSERT INTO 语句向回复信息表中添加回复信息，在页面中实现回复成功的提示信息，并将网页重定向到查看主题信息页面，程序代码如下：

```
<jsp:useBean id="connection" scope="request" class="com.JDBConnection"/>
<jsp:useBean id="showTime" scope="request" class="com.ShowTime"/>            //应用 JavaBean
<%
request.setCharacterEncoding("gbk");                                         //设置请求编码
String title=request.getParameter("title");                                  //获取回复主题信息
String content=request.getParameter("content");                              //获取回复内容信息
String id=request.getParameter("id");
String account=(String)session.getAttribute("account");                      //获取 session 范围内对象
String sql="insert into tb_forumBack values ('"+title+"','"+content+"','"+id+"','"+account+"','"+showTime.showTodayTime()+"')";
%>
connection.executeUpdate(sql);                                               //执行 SQL 语句
connection.closeConnection();                                                //关闭连接
```

秘笈心法

回复主题信息时，首先需要获取主题信息的 id，然后将主题 id 和回复信息插入到回复信息表中，这样在查询主题信息时，可以根据主题信息的 id 查询该主题相应的回复信息。

实例 589　删除主题及回复信息

光盘位置：光盘\MR\23\589

初级

实用指数：★★★

实例说明

版主在论坛中具有最高权限，版主登录后，可以对没有价值的主题进行删除操作。如果版主是该主题的发布者，那么版主同样拥有删除此主题的权利。在开发专业论坛网站的过程中，版主拥有删除论坛主题及回复信息的权利。BBS 的版块有公共和私有之分。公共的版块是任何登录本系统的用户都能看到的，私有的版块则只有指定的成员能够进入。运行本实例，以版主身份登录论坛，在查看主题信息页面时，“删除主题”和“删除回复”超链接处于可用状态。单击“删除主题”超链接的同时会删除与该主题绑定的回复信息，单击“删除回复”超链接直接删除回复信息。运行结果如图 23.14 所示。

关键技术

版主登录论坛后有删除主题信息以及删除回复主题信息的权利，主题信息表（tb_forumSend）和回复主题表（tb_forumBack）存在主键表和外键表的关系，关系图如图 23.15 所示。



图 23.14　删除主题及回复信息

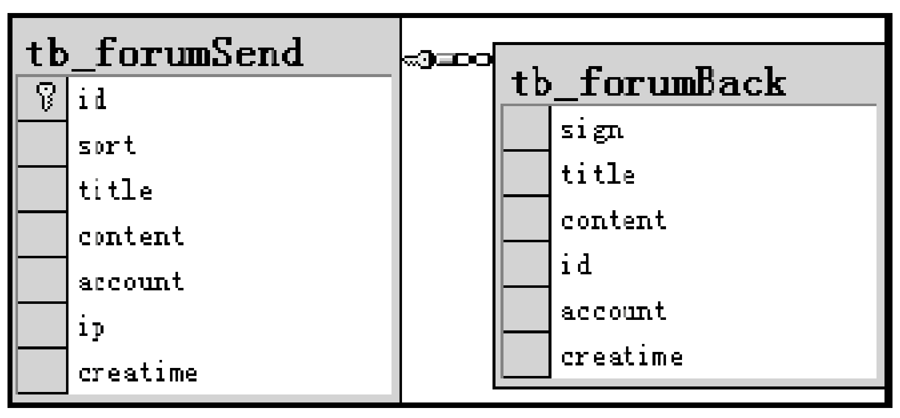


图 23.15　主题信息表和回复主题表的关系图

设计过程

（1）删除回复信息的程序代码如下：

```
<jsp:useBean id="connection" scope="request" class="com.JDBConnection"/>        //应用 JavaBean
<%
String sign=request.getParameter("sign");                                          //获取请求参数
String id=request.getParameter("id");
String sql="delete from tb_forumBack where sign='"+sign+"'";                       //执行删除操作的 SQL 语句
connection.executeUpdate(sql);                                                     //执行 SQL 语句
connection.closeConnection();                                                      //关闭连接
%>
<script language="javascript" type="text/javascript">
window.location.href='showContent.jsp?id=<%=id%>';
</script>
```

（2）删除主题信息的程序代码如下：

```
<jsp:useBean id="connection" scope="request" class="com.JDBConnection"/>        //应用 JavaBean
<%
String id=request.getParameter("id");                                              //获取请求参数
String sqlBack="delete from tb_forumBack where id='"+id+"'";                       //删除操作的 SQL 语句
connection.executeUpdate(sqlBack);                                                 //执行 SQL 语句
String sqlSend="delete from tb_forumSend where id='"+id+"'";
connection.executeUpdate(sqlSend);
connection.closeConnection();                                                      //关闭连接
%>
<script language="javascript" type="text/javascript">
window.location.href='showInformation.jsp';
</script>
```

秘笈心法

由于主题信息表与回复主题表是存在外键关联的，所以在删除主题信息表某个主题信息时，需要同时删除回复主题表中对应的回复信息。

实例 590	注销用户 光盘位置：光盘\MR\23\590	初级 实用指数：★★★

实例说明

进入本论坛后，如果用户没有进行注册或登录，那么此时用户为游客身份。而本实例只有成功注册后才可以回复留言信息，当游客单击“回复主题”超链接时，将会重定向到系统首页。为方便用户注册，在首页中提供了用户注册以及登录的功能。单击“用户注册”超链接可以注册一个新的用户。单击“登录”超链接，以注册的用户身份登录本论坛，从而具备删除主题及回复信息权限（只有管理员具备该权限）以外的所有权限，并享有一个离线的功能，即“注销用户”功能。运行本实例，以注册用户的身份登录本论坛进行发表、查看或回复主题等操作。单击“注销用户”超链接，即可以游客的身份重新定向到论坛首页。如果没有安全退出，非注册用户也可以通过当地浏览器以当前用户的身份进入本论坛。运行结果如图 23.16 所示。



图 23.16　注销用户

关键技术

利用 JSP 中的内置对象 session，只要客户端 session 变量有数据，这些保存的数值就可以被调用。通常，session 变量将在用户最后一次请求页面后保持 20 分钟的时间。为了避免闲置的 session 变量消耗计算机资源并防止他人盗取用户信息，程序提供了一个退出的功能，让用户在关闭窗口之前，先清除已经存在的 session 变量。

设计过程

利用 session.invalidate()语句清除 session 对象，并释放其所占用的资源，然后重新定向到系统首页，程序代码如下：

```
<%
        session.invalidate();                                  //清除 session 对象
        response.sendRedirect("index.jsp");                    //请求重定向地址
%>
```

秘笈心法

在用户登录时，将用户对象存放在 session 中。所以在用户退出系统时，应该调用 session.invalidate()语句清除 session 对象。

23.4 购 物 车

电子商务系统中的购物车同实际生活中的购物车一样，都是方便用户暂时保存挑选的商品的。购物车程序主要包括添加所选商品、查看购物车、修改单件商品购买数量、从购物车中移去指定商品和清空购物车 5 个部分。用户登录后，单击商品展台中的“购买”按钮，可以将对应的商品添加至购物车，购物车中将保存商品的 id 号、商品名称、单价、购买数量、单种商品的金额以及购物车内全部商品的合计金额。在查看购物车页面中，在“数量”文本框中输入购买数量后，单击“修改”按钮即可修改指定商品的购买数量；单击“清空购物车”超链接，将退回购物车中的全部商品；如果用户确定购买当前购物车中的全部商品，可以单击“去收银台结账”超链接进行订单处理。

实例 591	添加至购物车 光盘位置：光盘\MR\23\591	初级 实用指数：★★★

实例说明

添加至购物车页面主要用于将商品信息暂时保存到购物车中，在一些商品进销存网站中，“添加至购物车”是客户端程序中一个非常关键的功能，主要用来帮助用户完成商品的选购。单击“购买”按钮即可将商品添加到购物车中。运行结果如图 23.17 所示。



图 23.17 添加至购物车

关键技术

将商品信息添加至购物车，可以分为以下两种情况：

（1）当 shop 为空时，也就是当用户每次向购物车中添加第一件商品时需要新建一个 shop，然后将商品信息保存到 shop 中。

（2）当 shop 不为空时，说明购物车中已经有选购的商品了，这时不需要新建 shop，直接向购物车中添加商品信息即可。如果商品重复，可修改 shop 中的商品数量。

设计过程

（1）创建 BuyList.java 类文件，它通过属性对象保存购物车中的商品信息，具体代码如下：

```
public class BuyList {
    public String warename;
    public float price;
    public int number;
    public String photo;
    public String id;
}
```

（2）创建 addProduction.jsp 页面文件。

首先，通过 session 判断客户端用户是否已经购物，如果还没有购买商品，则首先将商品信息（商品 id、商品价格等信息）存储到 BuyList 类的对象属性中，使用 session 保存该对象。如果商品已经被购买，则要将 session 参数内的购物信息赋给新的 BuyList 类对象。其次，判断用户当前购买的商品是否已经存在，如果存在则将该商品的购物数量加 1，如果不存在，则在数组中添加新的数据。最后再赋给 session 参数，主要程序代码如下：

```
<%@page contentType="text/html; charset=GBK" language="java" import="java.sql.*" errorPage=""%>
<%@page import="com.dao.BuyList"%>
<%@page import="java.util.*"%>
<%
   request.setCharacterEncoding("GBK");
   String name = request.getParameter("name");
   String price = request.getParameter("price");
   String photo = request.getParameter("photo");
   String id=request.getParameter("id");
   //以上代码是从页面中取出数据
   BuyList buyList = new BuyList();
   buyList.id=id;
   buyList.number = 1;
   buyList.photo = photo;
   buyList.price = Float.parseFloat(price);
   buyList.warename = name;
   boolean flag = true;
   //以上代码是向 BuyList 类中的属性赋值
   List shoplist=null;
if (session.getAttribute("shop")==null){      //判断 session 中是否存在 shop 对象
    shoplist=new ArrayList();
}else{
   shoplist=(List)session.getAttribute("shop");
for(int i=0;i<shoplist.size();i++){
    BuyList buyitem=(BuyList)shoplist.get(i);
    if(buyitem.id.equals(buyList.id)){       //判断购物车中是否已经存在商品
    buyitem.number++;
    shoplist.set(i,buyitem);
    flag=false;
        }
    }
}
if(flag)shoplist.add(buyList);
session.setAttribute("shop",shoplist);
%>
<script language="javascript" type="text/javascript">
alert("商品已成功添加到购物车");
window.location.href="index.jsp";
</script>
```

秘笈心法

本实例主要是用 session 保存用户的商品信息，首先判断 session 中是否存在商品列表的 List 集合，如果不存在，则创建一个新的 List 集合；如果存在，则判断此时添加的商品是否存在于 List 集合；如果集合中包含此商品，则不添加，否则将这个商品添加到 List 集合中，最后将 List 集合保存在 session 中。

实例 592　查看购物车

光盘位置：光盘\MR\23\592

初级

实用指数：★★★

实例说明

为了方便用户随时查看购物车的情况，在添加至购物车页面中单击“购买”按钮，即可打开“购物车”页面。通过该页面可以查看已经放入购物车内的所有商品信息，运行结果如图 23.18 所示。



图 23.18　查看购物车

关键技术

查看购物车主要是将存储到 session 中的购物信息显示到页面中，实现该功能的程序代码如下：

```
<%
    List shopList=(List)session.getAttribute("shop");
      if(shopList==null||shopList.size()==0){
    }
%>
```

设计过程

在查看购物车时，首先需要查看购物车是否为空。如果为空，则需要将页面重定向到购物车首页面，否则显示购物车信息，将保存在 session 中的购物车信息利用 for 循环语句输出到浏览器中，其关键代码如下：

```
<form name="form" method="post" action="addProductionNumber.jsp">
    <table width="562" border="1" align="center" cellpadding="0" cellspacing="0">
      <tr align="center" bgcolor="#EFEFEF">
        <td width="39" height="22">序号</td><td width="112">商品名称</td>
        <td width="98">价格</td><td width="112">数量</td>
        <td width="109">总金额</td>
      </tr>
    <%
    List shopList= (List) session.getAttribute("shop");
      if (shopList==null||shopList.size()==0) {
    %>
<script language="javascript" type="text/javascript">
   alert("购物车中没有物品");
    window.location.href="index.jsp";
</script>
    <%
      } else {
         float num=0;
         int pric=0;
         for (int i = 0; i < shopList.size(); i++) {
            BuyList shop = (BuyList) shopList.get(i);
              num=num+shop.number*shop.price;
    %>
<script language="javascript" type="text/javascript">
function check(){
if(isNaN(form.number<%=i%>.value)){
alert("请不要输入非法字符");
return false;
history.back();
}
if(form.number<%=i%>.value==""){
alert("请输入修改的数量");
```

```
return false;
history.back();
}
}
</script>
      <tr align="center" bgcolor="#EFEFEF">
         <td height="21"><%=i+1%></td>
         <td><%=shop.warename%></td>
         <td><%=shop.price%><input name="id<%=i%>" type="hidden" size="10" value="<%=shop.id%>"></td>
         <td><input name="number<%=i%>" type="text" size="10" value="<%=shop.number%>"></td>
         <td><%=shop.number*shop.price%></td>
      </tr> <%}%>
  </table>
   <table width="562" border="0" align="center" cellpadding="0" cellspacing="0">
      <tr align="right">
         <td height="31" bgcolor="#EFEFEF">总合计金额：<%=num%></td>
      </tr>
  </table>
    <table width="303" border="0" align="center">
      <tr align="center" bgcolor="#EFEFEF">
      <td width="61" height="27">
         <input type="submit" name="Submit" value="修改数量" onClick="return check()">     </td>
         <td width="75"><a href="index.jsp">继续购物</a></td>
         <td width="74"><a href="putin.jsp">清空购物车</a></td>
         <td width="75"><a href="checkout.jsp">去收银台</a></td>
        </tr>
   </table> </form>
   <%}%>
```

秘笈心法

查看购物车，主要是从 session 中取出商品列表的 List 集合，然后循环这个集合，将所有商品信息显示在页面中即可。

实例 593　修改商品购买数量及从购物车中移除指定商品

光盘位置：光盘\MR\23\593

初级

实用指数：★★★

实例说明

为了满足用户的不同需要，购物车中还需要加入指定商品购买数量的功能。在购物车中，由于商品的数量被存放在文本框中，用户只需要在某种商品后面的“数量”文本框中输入相应的数量，单击“修改数量”按钮即可修改数量。当在“数量”文本框中输入数字 0 时，则为移去该商品。程序运行结果如图 23.19 所示。



图 23.19　修改商品购买数量及从购物车中移去指定商品

关键技术

当在文本框中输入 0 时，则是从购物车中移去该商品。移去该商品的程序代码如下：

```
list.remove(buyList);
```

list 是 List 类的对象，它存在于 java.util.*包中。

设计过程

单击“修改数量”按钮触发的是 addProductionNumber.jsp 页面，该页面主要实现修改商品数量的功能，主

要程序代码如下：

```
<%@ page contentType="text/html; charset=GBK" language="java" import="java.sql.*" errorPage="" %>
<%@ page import="java.util.*"%>
<%@ page import="com.dao.BuyList"%>
<%
List list=(List)session.getAttribute("shop");
for(int i=0;i<list.size();i++){
    String number=request.getParameter("number"+i);     //通过 for 循环取得文本框的数据
        int changeNumber=Integer.parseInt(number);
 if(changeNumber<0){                                    //如果取出的数据小于 0，则返回上一页面
%>
<script language="javascript" type="">history.back();</script>
 <%
 } else{
        String id=request.getParameter("id"+i);
  BuyList buyList=(BuyList)list.get(i);
        if(id.equals(buyList.id)){
  try{
        int newnum=Integer.parseInt(request.getParameter("number"+i));
  if(newnum!=0){
        buyList.number=newnum;
  }else{
  list.remove(buyList);
  }
        }catch(Exception e){
out.println("<script language='javascript'>alert('您输入的数量不是有效的整数!');history.back();</script>");
       return;
  }
}}}
session.setAttribute("shop",list);
response.sendRedirect("showShopping.jsp");
%>
```

秘笈心法

从购物车中移除商品，也就是从 List 集合中移除某个元素，首先从 session 中取出购物车的 List 集合，然后再根据用户选择的要移除的商品，从 List 集合中移除相对应的元素。

实例 594	清空购物车 光盘位置：光盘\MR\23\594	初级 实用指数：★★★

实例说明

在实例 593 中已经介绍了如何从购物车中移除指定的商品，下面将介绍一种将购物车中的全部商品一次性清空的方法，即清空购物车。清空购物车的实现方法很简单，只需要将保存在 session 中的购物信息清空即可。运行结果如图 23.20 所示。



图 23.20　清空购物车

关键技术

除了设定 session 变量的生命周期，用户暂停浏览动作超过一定时间后，变量中的存储值就会自动消失外，还可以使用 invalidate()方法将所有的 session 变量清除。本实例应用 invalidate()方法，语法如下：

```
session.invalidate()
```

其中，invalidate()用于销毁 session 对象。

设计过程

将 session 中的购物信息清空的程序代码如下：

```
<% session.invalidate()%>
```

秘笈心法

清空购物车，也就是将 session 对象中保存的 List 集合清除，应用 session 对象的 invalidate()方法可以将所有的 session 变量清除。

实例 595　收银台结账

光盘位置：光盘\MR\23\595

高级

实用指数：★★★

实例说明

如同在超市中购物一样，将商品保存到购物车中并不是网上购物的最终目的，而到收银台结账后，才算一次购物的最终完成。前面所有功能都是为最后生成一个用户满意的订单做准备。生成订单时，不仅要保存用户订单中所购买的商品信息和订单信息，同时还需要返回一个可供用户随时查询的订单号。用户单击查看购物车页面中的“去收银台”超链接即可进入到收银台结账页面输入订单信息，订单信息输入完成后，单击“提交”按钮即可保存订单信息到数据表中。运行结果如图 23.21 所示。



图 23.21　收银台结账

关键技术

在收银台结账页面中，首先应该判断用户是否已经购物，然后判断用户是否已经登录，如果用户没有购物或没有登录都将给予提示并返回到网站首页，否则显示输入订单信息的表单。

用户在收银台页面输入订单信息后，单击“提交”按钮将进入到保存订单页面，将订单信息分别保存到订单主表和订单明细表中。保存订单信息页面可以通过以下步骤实现：

（1）为保存订单信息做准备。判断购物车是否为空，判断用户是否已经登录。

（2）获取订单信息。首先通过 session.getAttribute("shop")获取商品信息，然后保存到订单主表和明细表中。

设计过程

单击“提交”按钮将进入到保存订单页面。首先将订单的概要信息保存到订单主表中，同时返回该订单编号，然后通过 for 循环语句插入订单明细表信息，关键代码如下：

```
<%@page contentType="text/html; charset=GBK" language="java" import="java.sql.*" errorPage=""%>
<%@page import="com.dao.BuyList"%>
<%@page import="java.util.*"%>
<jsp:useBean id="connection" scope="request" class="com.tool.JDBConnection"/>
<%
  request.setCharacterEncoding("GBK");
  String orderNumber = request.getParameter("orderNumber");
  String name = request.getParameter("name");
  String realName = request.getParameter("realName");
  String email = request.getParameter("email");
  String address = request.getParameter("address");
  String tel = request.getParameter("tel");
  String post = request.getParameter("post");
  String sqlOrderMan = "insert into tb_shop_orderMan values('" + orderNumber + "','" + name + "','" + realName + "','" + email + "','" + post + "','" + tel
+ "','" + address + "')";
  connection.executeUpdate(sqlOrderMan);
  List list = (List) session.getAttribute("shop");
  for (int i = 0; i < list.size(); i++) {
    BuyList buy = (BuyList) list.get(i);
    String sqlOrderPr = "insert into tb_shop_orderProduction values ('" + orderNumber + "','" + buy.warename + "','" + buy.id + "','" + buy.photo + "','" +
buy.price + "','" + buy.number + "')";
      connection.executeUpdate(sqlOrderPr);
```

```
    }
    session.invalidate();
    connection.closeDBConnection();
    response.sendRedirect("index.jsp");
%>
```

秘笈心法

收银台结账，主要就是将用户的联系信息作为订单，保存到数据库中即可。

23.5　聊　天　室

随着互联网的飞速发展，聊天室这种比较古老的交流方式已经被众多人所认可。通过聊天室在线聊天已成为网络上人与人之间沟通、交流和联系的一种方式。为此，越来越多的网站开始提供在线聊天的功能。下面通过几个实例介绍如何设计不同风格、不同功能的聊天室。

实例 596	Application 形式的聊天室 光盘位置：光盘\MR\23\596	高级 实用指数：★★★

实例说明

说到聊天室，大家都不会陌生，相信许多人都有过上网聊天的经历，那么该如何制作自己的聊天室呢？本实例主要应用 Application 对象实现聊天室程序。运行程序，首先登录聊天室，在“昵称”文本框中输入用户登录的昵称，单击“我要聊天”按钮进入聊天室，此时就可以看到聊天室中所有人的相关信息。程序运行结果如图 23.22 和图 23.23 所示。



图 23.22　聊天室登录页面



图 23.23　聊天室主界面

关键技术

在本实例中主要应用 Application 对象的 3 个方法。

- ❑ application.getAttribute(String name)：返回由 name 指定名字的 Application 对象的属性值。
- ❑ application.setAttribute(String name,Object object)：设置由 name 指定名字的 Application 对象的属性值 object。
- ❑ application.removeAttribute("")：清除 Application 指定对象。

设计过程

（1）当用户输入自己的昵称后，单击“我要聊天”按钮直接进入聊天室的页面，如图 23.23 所示。该页面主要由 4 部分组成，上侧页面（left_top.jsp）、显示聊天内容的页面（left.jsp）、发送聊天信息的页面（down.jsp）以及显示在线聊友的页面（right.jsp），这些页面的导入代码在 main.jsp 页面中，该页面的代码如下：

```
<frameset rows="*" cols="1*,1024,1*" framespacing="0" frameborder="no" border="0">
<frame src="blank.htm" name="BLFrame" scrolling="no" noresize="noresize">
<frameset rows="*" cols="*,182" framespacing="0" frameborder="no" border="0">
  <frameset rows="*,95" cols="*" framespacing="0" frameborder="no" border="0">
  <frameset rows="103,*" cols="*" framespacing="0" frameborder="no" border="0">
    <frame src="left_top.jsp" name="leftTopFrame">
<frame src="left.jsp" name="mainFrame">
</frameset>
    <frame src="down.jsp" name="bottomFrame" scrolling="NO" noresize>
  </frameset>
  <frame src="right.jsp" name="rightFrame" scrolling="NO" noresize></frameset>
 <frame src="blank.htm" name="BLFrame" scrolling="no" noresize="noresize">
 </frameset>
<noframes>
```

（2）显示聊天内容的页面（left.jsp），主要通过 JavaScript 代码实现刷新页面，并分行显示聊天的记录，关键代码如下：

```
<script language="JavaScript" type="text/javascript">
function GetData(url){
url="dealWith.jsp?action=showMessage";                          //调用页面
try{
 Load.src = url;
 window.location.href="left.jsp#bottom";                        //刷新页面到底部
}catch(e){
 return false;
}
var timeoutid = setTimeout("GetData()",2000);                   //每隔 2 秒调用一次 GetData()函数
}
</script>
<script id=" Load " language="JavaScript" type="text/javascript" defer></script>
</head>
<body onLoad="GetData();" bgcolor="#FFEBB8">
<span id="loadContent"><br>欢迎来到本聊天室！</span>
<a name="bottom"> </a>
```

（3）发送聊天信息的页面（down.jsp），主要实现用户在文本框中输入内容后，单击“发送”按钮将信息传递到 left.jsp 页面，关键代码如下：

```
<script language="javascript">
function check(form1){
 if (form1.content.value==""){alert("请输入内容！");return false;}    //判断是否输入聊天内容
}
</script>
<form name="form1" method="post" action="dealWith.jsp?action=sendMessage" onSubmit="return check(form1)">
  <table width="702" border="0" align="center" cellpadding="0" cellspacing="0">
    <tr >
      <td width="101" align="right"><%=session.getAttribute("username") %>说： </td>
      <td width="434"><input name="content" type="text" size="70">
      </td>
      <td width="167"><input name="Submit" type="submit" class="btn_grey" value="发送">
         
        <input name="Submit2" type="button" class="btn_grey"
           onClick="parent.location.href='dealWith.jsp?action=loginOut';" value="退出聊天室">
      </td>
    </tr>
  </table>
</form>
```

（4）显示在线人数的页面（right.jsp）的关键代码如下：

```
<script language="JavaScript">
function GetData(url){
 url="dealWith.jsp?action=showMsg";                             //调用页面
 try{
        DataLoad.src = url;
 }catch(e){
       return false;
 }
 var timeoutid = setTimeout("GetData()",2000);                  //每隔 2 秒调用一次 GetDate()函数
}
```

```
</script>
<script  id="DataLoad" language="JavaScript" type="text/javascript" defer ></script>
</head>
<body onLoad="GetData();"   background="images/right.jpg">
<div align="center"> 所有人 <span id="loadContent">数据加载中……</span> </div>
<%if(session.getAttribute("username")==null){
  out.println("<script language='javascript'>parent.location.href='index.jsp';</script>");
}%>
</body>
```

（5）业务处理页面（dealWith.jsp）是整个聊天室的关键，它通过 if 语句判断 action 参数值来进行不同的操作，关键代码如下：

```
<%@page contentType="text/html; charset=GBK" language="java"import="java.util.*" errorPage=""%>
<%!String msg;%>
<%
request.setCharacterEncoding("GBK");
String action = request.getParameter("action");                    //获取 action 参数值
boolean flag = true;
if (action.equals("login")) {                                      //用户登录
        application.setAttribute("flag", "true");
        String username = request.getParameter("username");
        List userList = (List) application.getAttribute("userList");   //从 Application 中获取用户列表
        if (application.getAttribute("userList") == null) {
                userList = new ArrayList();
        } else {
                for (int i = 0; i < userList.size(); i++) {            //通过循环判断登录的用户名是否存在
                        String user = (String) userList.get(i);        //获取一个用户
                        if (user.equals(username)) {
                                flag = false;
                        }
                }
        }
        //以上代码判断用户是否在线
        if (flag) {
                userList.add(username);
                application.setAttribute("userList", userList);        //保存用户列表到 session 中
                session.setAttribute("username", username);            //保存用户名到 session 中
                response.sendRedirect("main.jsp");                     //将页面跳转到聊天室主界面
        } else {
                out.println("<script language='javascript'>parent.location.href='index.jsp?result=1';</script>");
        }
}
if (action.equals("loginOut")) {                                   //离开聊天室
        application.setAttribute("flag", "false");
        String username = (String) session.getAttribute("username");
        List userList = (List) application.getAttribute("userList");
        for (int i = 0; i < userList.size(); i++) {
                String user = (String) userList.get(i);
                if (user.equals(username)) {
                        userList.remove(i);                            //从用户列表中移除指定用户
                        session.invalidate();                          //清空 session
                        out.println("<script language='javascript'>parent.location.href='index.jsp';</script>");
                }
                if (userList.size() == 0) {
                        application.removeAttribute("message");        //从 Application 中删除聊天记录
                }
        }
        response.sendRedirect("index.jsp");
}
if (action.equals("showMsg")) {                                    //查询在线人数
        String mm = (String) application.getAttribute("flag");
        String username = "";
        if (application.getAttribute("userList") != null) {
                List userList = (List) application.getAttribute("userList");  //从 Application 中获取用户列表
                for (int i = 0; i < userList.size(); i++) {
                        String name = (String) userList.get(i);
                        username = username + "<br><br>" + name;       //组合在线用户列表
                }
```

```
        }
        application.setAttribute("flag", "false");
        out.println("loadContent.innerHTML=\"" + username + "\";");          //显示在线用户列表
}
if (action.equals("sendMessage")) {                                           //发表信息
        if (msg == null) {
                msg = "<br>欢迎进入聊天室";
        }
        msg = (String) msg + "<br>" + session.getAttribute("username")
                        + ":" + (String) request.getParameter("content");     //组合聊天内容
        response.sendRedirect("down.jsp");                                    //将页面重定向到 down.jsp 页面
}
if (action.equals("showMessage")) {
        if (msg != null) {
                out.println("loadContent.innerHTML=\"" + msg + "\";");        //显示聊天内容列表
        }
}
%>
```

秘笈心法

由于该聊天室是应用 Application 对象实现的，所以聊天信息是保存在 Application 对象中的。利用 Application 对象保存信息时，当服务器重新启动后，Application 对象中的数据将全部清空。这个是 Application 对象的最大特点。

实例 597	带私聊的聊天室 光盘位置：光盘\MR\23\597	高级 实用指数：★★★

实例说明

通常的聊天室都会带有私聊功能。所谓私聊就是私聊信息只有发言人和接收人可以看到该信息，其他用户不能看到该信息，这样可以保证用户间的悄悄话不被其他人看到。本实例将介绍通过 JSP+Ajax 实现带私聊功能的聊天室。运行本实例，在用户登录页面中输入用户昵称，单击“登录”按钮，可以进入到聊天室的主界面，在该页面中，用户之间可以进行交流，如果想要与某个用户进行私聊，可以选中“悄悄话”复选框。实例运行结果如图 23.24 所示。

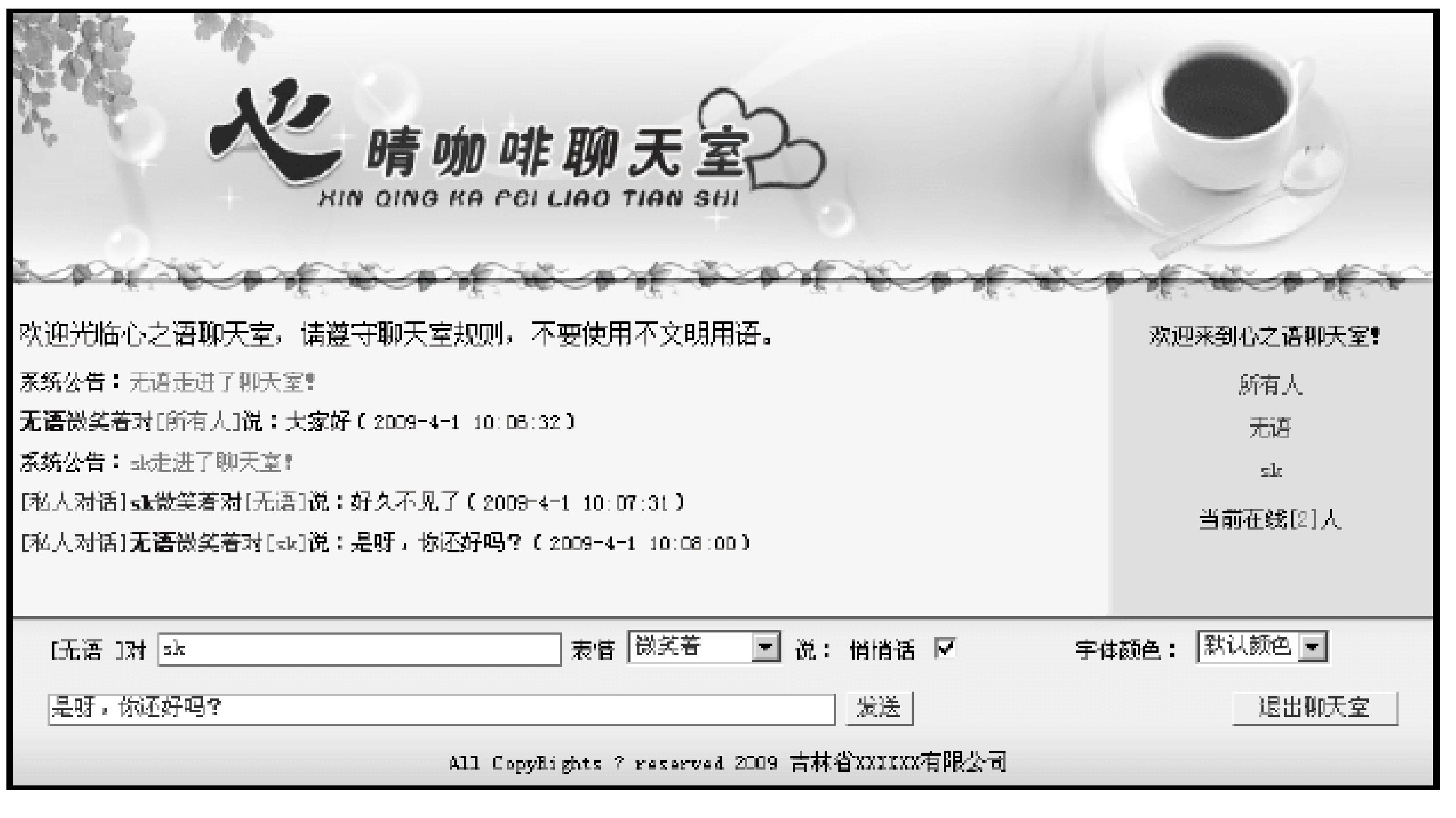


图 23.24　聊天室主界面

关键技术

本实例中主要应用的技术是 Ajax 重构、Application 对象和 Vector 类的相关方法。关于 Ajax 重构的方法请参见第 10 章，Application 对象的详细介绍请参见实例 596。下面对 Vector 类进行详细介绍。

Vector 类是一元集合，可以加入重复数据，它的作用和数组类似，可以保存一系列数据，它的优点是可以

很方便地对集合内的数据进行查找、增加、修改和删除等操作。下面介绍 Vector 类的常用方法。

- add(int index,Object element)：在指定的位置添加元素。
- addElementAt(Object obj,int index)：在 Vector 类的结尾添加元素。
- size()：返回 Vector 类的元素总数。
- elementAt(int index)：取得特定位置的元素，返回值为整型。
- setElementAt(object obj,int index)：重新设定指定位置的元素。
- removeElementAt(int index)：删除指定位置的元素。

设计过程

（1）实现登录聊天室。

编写聊天室登录页面 index.jsp。该页面主要用于收集用户输入的登录信息，以及通过自定义的 JavaScript 函数验证输入信息是否为空和是否包括非法字符，该页面的表单元素只包括输入用户名的文本框和提交表单的按钮，具体代码如下：

```
<form name="form1" method="post" action="Messages?action=loginRoom" onSubmit="return check()">
用户名：<input type="text" name="username" class="login">
<input name="Submit" type="submit" class="btn_bg" value="进  入">
</form>
```

编写用来保存在线用户和对在线用户进行具体操作的类，这里为 UserInfo，在该类中主要包括返回外界使用的实例对象、添加用户、获取用户列表和移除用户的方法。UserInfo 类的具体代码如下：

```
package com.wgh.model;
import java.util.Vector;
public class UserInfo {
 private static UserInfo user = new UserInfo();
 private Vector vector = null;
 //利用 private 调用构造函数，防止被外界产生新的 instance 对象
 private UserInfo() {
        this.vector = new Vector();
 }
 //外界使用的 instance 对象
 public static UserInfo getInstance() {
        return user;
 }
 //增加用户
 public boolean addUser(String user) {
        if (user != null) {
               this.vector.add(user);
               return true;
        } else {
               return false;
        }
 }
 //获取用户列表
 public Vector getList() {
        return vector;
 }
 //移除用户
 public void removeUser(String user) {
        if (user != null) {
               vector.removeElement(user);
        }
 }
}
```

创建 UserListener 类，主要实现 valueBound(HttpSessionBindingEvent arg0)和 valueUnbound(HttpSessionBindingEvent arg0)方法，用于监控 session 中的对象变化情况。在 valueBound()方法中向控制台输出上线用户的信息，在 valueUnbound()方法中向控制台输出下线用户的信息。UserListener 类的具体代码如下：

```
package com.wgh.servlet;
import com.wgh.model.UserInfo;
import javax.servlet.http.HttpSessionBindingEvent;
```

```
public class UserListener implements
        javax.servlet.http.HttpSessionBindingListener {
private String user;
private UserInfo container = UserInfo.getInstance();     //获得 UserInfo 类的对象
public UserListener() {
        user = "";
}
//设置在线监听人员
public void setUser(String user) {
        this.user = user;
}
//获取在线监听
public String getUser() {
        return this.user;
}
//当 session 有对象加入时执行的方法
public void valueBound(HttpSessionBindingEvent arg0) {
        System.out.println("上线用户：" + this.user);
}
//当 session 有对象移除时执行的方法
public void valueUnbound(HttpSessionBindingEvent arg0) {
        System.out.println("下线用户：" + this.user);
        if (user != "") {
                container.removeUser(user);
        }
}
}
```

编写聊天室的 Servlet 实现类，并在该类中添加登录聊天室的方法 loginRoom()，在该方法中，首先获取登录用户，然后判断该用户是否登录。如果已经登录，给出提示信息，否则将该用户添加到在线用户列表中，并且向 Application 对象的聊天内容属性中添加一条系统公告信息，最后将页面重定向到登录成功页。loginRoom()方法的具体代码如下：

```
public void loginRoom(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
response.setContentType("text/html;charset=GBK");
request.setCharacterEncoding("GBK");
ChStr chStr=new ChStr();
HttpSession session = request.getSession();
String username=chStr.toGBK(request.getParameter("username"));          //获得登录用户名
UserInfo user=UserInfo.getInstance();                                   //获得 UserInfo 类的对象
session.setMaxInactiveInterval(600);                                    //设置 session 的过期时间为 10 分钟
Vector vector=user.getList();
boolean flag=true;                                                      //标记是否登录的变量
//判断用户是否登录
if(vector!=null&&vector.size()>0){
        for(int i=0;i<vector.size();i++){
                if(user.equals(vector.elementAt(i))){
                        PrintWriter out;
                        try {
                                out = response.getWriter();
                out.println("<script language='javascript'>alert('该用户已经登录');window.location.href='index.jsp';</script>");
                        } catch (IOException e) {
                                e.printStackTrace();
                        }
                        flag=false;
                        break;
                }
        }
}
//保存用户信息
if(flag){
        UserListener ul=new UserListener();                             //创建 UserListener 的对象
        ul.setUser(username);                                           //添加用户
        user.addUser(ul.getUser());                                     //添加用户到 UserInfo 类的对象中
        session.setAttribute("user",ul);                                //将 UserListener 对象绑定到 session 中
        session.setAttribute("username",username);                      //保存当前登录的用户名
        session.setAttribute("loginTime",new Date().toLocaleString());  //保存登录时间
```

```
}
    ServletContext application=getServletContext();
    Vector sourceMessage=null;
    if(null!=application.getAttribute("message")){
          sourceMessage=(Vector)application.getAttribute("message");
    }else{
          sourceMessage=new Vector();
    }
    MessageForm messageForm=new MessageForm();
    messageForm.setFrom("系统公告：");                                    //发言人
    messageForm.setTo("系统公告：");                                      //接收者
    messageForm.setContent("系统公告：<font color='gray'>" + username + "走进了聊天室！</font><br>"); //发言内容
    messageForm.setIsPrivate("false");                                    //是否为悄悄话
    sourceMessage.add(messageForm);
    application.setAttribute("message",sourceMessage);
    try {
          request.getRequestDispatcher("login_ok.jsp").forward(request, response);
    } catch (Exception ex) {
          Logger.getLogger(Messages.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

编写登录成功页 login_ok.jsp，在该文件中，将页面重定向到聊天室的主界面。登录成功页的具体代码如下：

```
<%@page contentType="text/html" pageEncoding="GBK"%>
<%response.sendRedirect("main.jsp");%>
```

（2）编写聊天室主界面 main.jsp。该页面共包括页面头部、聊天内容显示区、在线人员列表区和用户发言区 4 部分，具体布局如图 23.25 所示。

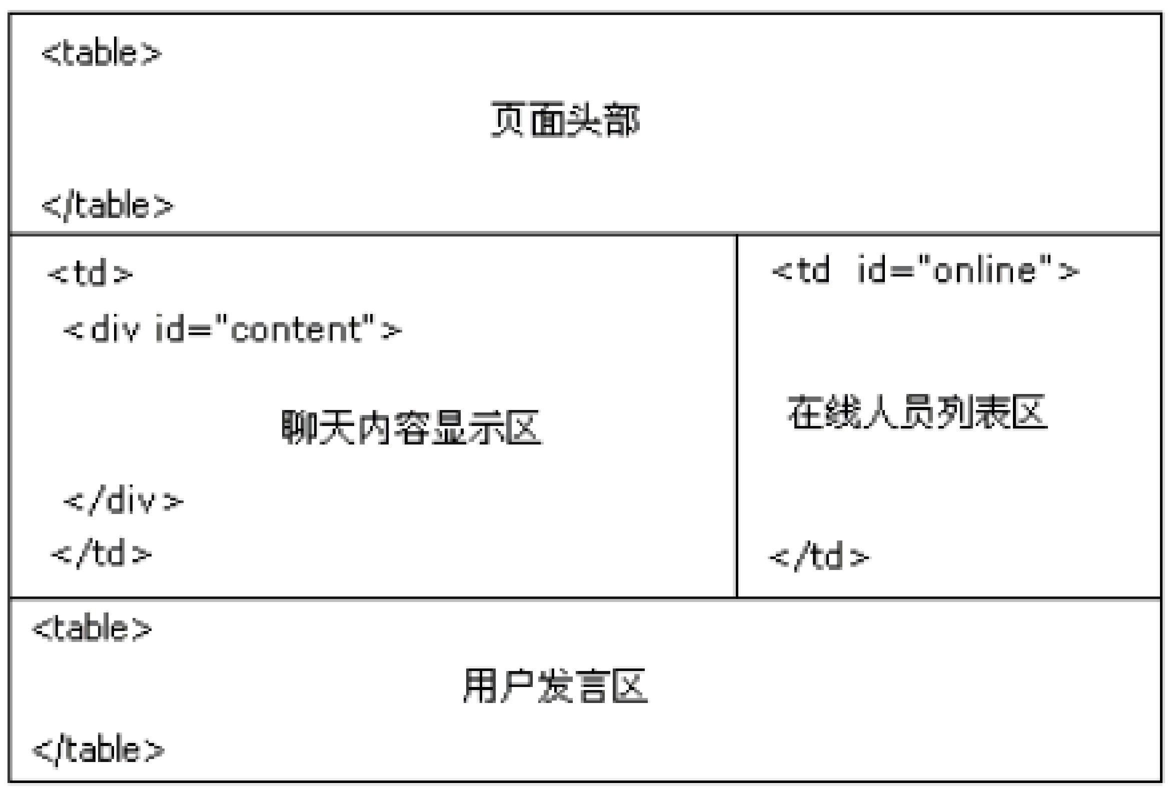


图 23.25　主界面的布局

布局后的主界面的关键代码如下：

```
……          <!--此处省略了页面头部的 HTML 代码-->
<table width="778" height="276" border="0" align="center" cellpadding="0" cellspacing="0">
  <tr>
    <td width="599"   height="200" valign="top" bgcolor="#fcf5eb" style="padding:5px; ">
<div style="height:290px; overflow:hidden" id="content">聊天内容</div>
</td>
    <td width="171" valign="top" background="images/right.jpg" id="online" style="padding:5px">
在线人员列表
</td>
  </tr>
</table>
<table width="778" height="96" border="0" align="center" cellpadding="0" cellspacing="0" bordercolor="#D6D3CE"
background="images/bottom.jpg">
……          <!--此处省略了用户发言区的代码，该代码将在实现用户发言时进行详细介绍-->
</table>
```

（3）重构 Ajax。关于 Ajax 重构的方法请参见第 10 章。

（4）实时获取并显示在线人员列表。

在 main.jsp 页面中，编写自定义的 JavaScript 函数 showOnline()，用于实例化 Ajax 对象。showOnline()函数的具体代码如下：

```
function showOnline(){
 var loader=new net.AjaxRequest("online.jsp?nocache="+
```

```
        new Date().getTime(),deal_online,onerror,"GET");
}
```

在上面的代码中，一定要加代码“?nocache="+new Date().getTime()”，否则将出现在线人员列表不更新的情况。

创建 online.jsp 文件。在该文件中，主要是将保存到集合类中的在线人员列表显示到页面。online.jsp 页面的代码如下：

```
<%@page contentType="text/html" pageEncoding="GBK" %>
<%@ page import="com.wgh.model.UserInfo"%>
<%@ page import="java.util.*"%>
<%
UserInfo list=UserInfo.getInstance();
Vector vector=list.getList();
int amount=0;
%>
<table width="100%" border="0" cellpadding="0" cellspacing="0">
  <tr><td height="32" align="center" class="word_orange ">欢迎来到心之语聊天室！</td></tr>
  <tr>
  <td height="23" align="center"><a   href="#" onclick="set('所有人')">所有人</a></td>
  </tr>
 <%if(vector!=null&&vector.size()>0){
 String username="";
 amount=vector.size();
        for(int i=0;i<amount;i++){
                username=(String)vector.elementAt(i);
        %>
  <tr>
     <td height="23" align="center"><a href="#" onclick="set('<%=username%>')"><%=username%></a></td>
  </tr>
<%}}%>
<tr><td height="30" align="center">当前在线[<font color="#FF6600"><%=amount%></font>]人</td></tr>
</table>
```

在聊天室的主界面中，将右侧用于显示在线人员列表的单元格的 id 属性设置为 online，用于实时显示在线人员列表，具体代码如下：

```
<td width="165" valign="top" bgcolor="#f6fded" id="online" style="padding:5px">在线人员列表</td>
```

编写 Ajax 的回调函数 deal_online()，用于将获取的在线人员列表赋值给 id 为 online 的<td>标记的 innerHTML 属性。deal_online()函数的具体代码如下：

```
function deal_online(){
 online.innerHTML=this.req.responseText;
}
```

为了让页面载入后就调用 Ajax 获取在线人员列表，并且每隔 10 秒钟便获取一次数据，还需要在页面中添加以下 JavaScript 代码：

```
window.setInterval("showOnline();",10000);
window.onload=function(){
 showOnline();                                    //当页面载入后显示在线人员列表
}
```

细心的读者可以发现，聊天对象文本框被设置为只读属性，这样用户就不能手动输入聊天对象，所以还需要提供选择聊天对象的功能，这可以通过在主页面中添加选择聊天对象的 JavaScript 自定义函数及在在线人员列表上添加超链接实现。实现将选择的聊天对象添加到聊天对象文本框的 JavaScript 代码如下：

```
<script language="javascript">
function set(selectPerson){  //自动添加聊天对象
 if(selectPerson!="${username}"){
                form1.to.value=selectPerson;
 }else{
        alert("请重新选择聊天对象！ ");
 }
}
</script>
```

在在线人员列表上添加超链接的具体代码如下：

```
<a href="#" onclick="set('<%=username%>')"><%=username%></a>
```

（5）实现用户发言。

在页面的合适位置添加用于收集用户发言信息的表单及表单元素，关键代码如下：

```
<form action="" name="form1" method="post" >
      <input name="from" type="hidden" value="${username}">[${username} ]对
         <input name="to" type="text" value="" size="35" readonly="readonly">
表情
<select name="face" class="wenbenkuang">
   <option   value="无表情的">无表情的</option>
   <option value="微笑着" selected>微笑着</option>
 <!--此处省略了添加其他列表项的代码-->
   <option value="无精打采的">无精打采的</option>
</select>
说：  悄悄话
<input name="isPrivate" type="checkbox" class="noborder" id="isPrivate" value="true" onClick="checkIsPrivate()"></td>
      <td width="189" align="left">   字体颜色：
         <select name="color" size="1" class="wenbenkuang" id="select">
            <option selected>默认颜色</option>
            <option style="color:#FF0000" value="FF0000">红色热情</option>
            <!—此处省略了添加其他列表项的代码-->
            <option style="color:#999999" value="999999">烟雨蒙蒙</option>
         </select>
      <input name="content1" type="text" size="70" onKeyDown="if(event.keyCode==13 && event.ctrlKey){send();}">
         <input name="Submit2" type="button" class="btn_grey" value="发送" onClick="send()">
      <input name="button_exit" type="button" class="btn_grey" value="退出聊天室" onClick="Exit()">
</form>
```

编写自定义的 JavaScript 函数 send()，用于调用 Ajax 实现用户发言。在该函数中，首先验证用户输入信息的合法性，然后再将提交的表单元素的内容连接为一个参数字符串，并实例化 Ajax 对象。send()函数的具体代码如下：

```
function send(){                                                                    //验证聊天信息并发送
 if(form1.to.value==""){
         alert("请选择聊天对象！");return false;
 }
 if(form1.content1.value==""){
         alert("发送信息不可以为空！");form1.content1.focus();return false;
 }
 if(form1.isPrivate.checked){                                                       //确定悄悄话变量的值
         isPrivate="true";
 }else{
         isPrivate="false";
 }
 var param="from="+form1.from.value+"&isPrivate="+isPrivate+"&face="+form1.face.value+"&color="+
         form1.color.value+"&to="+form1.to.value+"&content="+      form1.content1.value;
      var loader=new net.AjaxRequest("Messages?action=sendMessage",deal_send,onerror,"POST",param);
}
```

在聊天室相关的 Servlet 实现类中，添加发送聊天信息的方法 sendMessages()。在该方法中，首先获取并保存用户发言相关信息到 MessageForm 的对象 messageForm 中，在获取时需要对可能出现中文的信息进行转码，然后将 messageForm 保存到 Vector 中，最后将保存聊天内容的 Vector 保存到 Application 中，并重定向网页。sendMessages()方法的具体代码如下：

```
public void sendMessages(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
            response.setContentType("text/html;charset=GBK");
            request.setCharacterEncoding("GBK");
            ChStr chStr=new ChStr();
            Random random = new Random();
            MessageForm messageForm=new MessageForm();
            messageForm.setFrom(chStr.toUTF8(request.getParameter("from")));          //发言人
            messageForm.setFace(chStr.toUTF8(request.getParameter("face")));          //表情
            messageForm.setTo(chStr.toUTF8(request.getParameter("to")));              //接收者
            messageForm.setColor(request.getParameter("color"));                      //字体颜色
            messageForm.setContent(chStr.toUTF8(request.getParameter("content")));    //发言内容
            messageForm.setIsPrivate( request.getParameter("isPrivate"));             //是否为悄悄话
            messageForm.setSendTime( new Date().toLocaleString());                    //发言时间
            ServletContext application = getServletContext();
            Vector sourceMessage=(Vector)application.getAttribute("message");
            sourceMessage.add(messageForm);
```

```
            application.setAttribute("message", sourceMessage);                    //将聊天信息保存到 Application 中
            request.getRequestDispatcher("Messages?action=getMessages&nocache=" +
        random.nextInt(10000)).forward(request, response);                         //重定向页面
}
```

（6）实现实时显示聊天内容。

编写自定义的 JavaScript 函数 showContent()，用于实例化 Ajax 对象。showContent()函数的具体代码如下：

```
function showContent(){
 var loader1=new net.AjaxRequest("Messages?action=getMessages&nocache="+
        new Date().getTime(),deal_content,onerror,"GET");
}
```

在聊天室相关的 Servlet 实现类中，编写 getMessages()方法，用于将聊天室信息组合成字符串，并传递到显示聊天内容的 JSP 页面。在组合过程中，需要对私聊信息进行处理。getMessages()方法的具体代码如下：

```
public void getMessages(HttpServletRequest request,HttpServletResponse response)   throws ServletException, IOException {
      response.setContentType("text/html;charset=GBK");
      request.setCharacterEncoding("GBK");
      ServletContext application = getServletContext();
      Vector sourceMessage=(Vector)application.getAttribute("message");
      HttpSession session=request.getSession();
      String msg="";
      for(int i=0;i<sourceMessage.size();i++){
            MessageForm f=(MessageForm)sourceMessage.get(i);
            if("true".equals(f.getIsPrivate())){
                  String from=f.getFrom();
                  String to=f.getTo();
                  if(session.getAttribute("username").equals(to) || session.getAttribute("username").equals(from)){
                        msg += "<font color='red'>[私人对话]</font><font color='blue'><strong>" + f.getFrom() +
                 "</strong></font><font color='#CC0000'>" + f.getFace() + "</font>对<font color='green'>[" + f.getTo() +
                  "]</font>说：" + "<font color='" + f.getColor() + "'>" + f.getContent() + "</font>（" +
                  f.getSendTime() + "）<br>";
                  }
          }else{
                if(f.getFrom().equals(f.getTo()) && "系统公告：".equals(f.getFrom())){
                      msg += f.getContent();
                }else{
                      msg += "<font color='blue'><strong>" + f.getFrom() + "</strong></font><font color='#CC0000'>" +
                         f.getFace() + "</font>对<font color='green'>[" + f.getTo() + "]</font>说：" + "<font color='" +
                         f.getColor() + "'>" + f.getContent() + "</font>（" + f.getSendTime() + "）<br>";
                }
          }
     }
     request.setAttribute("msg", msg);                                              //保存聊天信息
     request.getRequestDispatcher("content.jsp").forward(request, response);        //重定向页面
}
```

编写显示聊天内容的 content.jsp 页面，在该页面中只需要应用 EL 表达式将返回的执行结果输出即可，具体代码如下：

```
<%@page contentType="text/html" pageEncoding="GBK" %>
${msg}
```

在聊天室的主界面中，在左侧用于显示聊天内容的单元格中，添加一个 id 属性为 content 的<div>标记，用于实时显示聊天内容，具体代码如下：

```
<div style="height:290px; overflow:hidden" id="content">聊天内容</div>
```

编写 Ajax 的回调函数 deal_content()，在该函数中，首先获取 Ajax 处理页的返回值，然后去除字符串中的 Unicode 空白符，最后判断在获取信息时是否产生错误，如果是，则退出聊天室，否则将获取的聊天内容赋值给 id 为 content 的<div>标记的 innerHTML 属性。deal_content()函数的具体代码如下：

```
function deal_content(){
 var returnValue=this.req.responseText;                                  //获取 Ajax 处理页的返回值
 var h=returnValue.replace(/\s/g,"");                                    //去除字符串中的 Unicode 空白符
 if(h=="error"){
        Exit();
 }else{
        content.innerHTML=sysBBS+returnValue+"</span>";
        //当聊天信息超过一屏时，设置最先发送的聊天信息不显示
```

```
        document.getElementById('content').scrollTop = document.getElementById('content').scrollHeight*2;
    }
}
```

为了让页面载入后就调用 Ajax 获取聊天内容，并且每隔 1 秒钟便获取一次数据，还需要在页面中添加以下 JavaScript 代码：

```
window.setInterval("showContent();",1000);
window.onload=function(){
 showContent();                                    //当页面载入后显示聊天内容
}
```

（7）实现退出聊天室。

编写自定义的 JavaScript 函数 Exit()，在该函数中首先将页面重定向到退出聊天室页面 leave.jsp，然后再弹出“欢迎您下次光临！”对话框，具体代码如下：

```
function Exit(){
 window.location.href="leave.jsp";
 alert("欢迎您下次光临！");
}
```

在“退出聊天室”按钮的 onClick 事件中调用自定义的 JavaScript 函数 Exit()，关键代码如下：

```
<input name="button_exit" type="button" class="btn_grey" value="退出聊天室" onClick="Exit()">
```

编写退出聊天室页面 leave.jsp，在该页面中，首先销毁 session，然后将页面重定向到登录页面。leave.jsp 页面的代码如下：

```
<%@page contentType="text/html" pageEncoding="GBK" %>
<%
session.invalidate();                              //清空 session
response.sendRedirect("index.jsp");                //重定向页面
%>
```

秘笈心法

HttpBindingListener 接口的 valueBound()方法在有对象加入 session 时会被自动执行，而 valueUnbound()方法则是有对象从 session 中移除时会被自动执行。

实例 598	XML 形式的聊天室 光盘位置：光盘\MR\23\598	高级 实用指数：★★★

实例说明

在实例 596 和实例 597 中，聊天记录均保存在 Application 对象中，这样当服务器关闭后，聊天记录将被自动删除。本实例将介绍一种将聊天记录保存到 XML 文件中的聊天室，这样聊天记录将不会被自动删除，方便以后进行查找。运行本实例，在用户登录页面中输入用户昵称，单击“登录”按钮，可以进入到聊天室的主界面，在该页面左侧的在线用户列表中，将显示最新的用户列表及在线人数；在右侧的聊天内容显示区中，将实时显示最新的聊天信息，当聊天信息超过一屏后，将自动滚屏；在下方的用户发言区，用户可以发送聊天信息。实例运行结果如图 23.26 所示。

关键技术

本实例中应用的主要技术是 Ajax 重构、Java 的文件操作类（java.io.File 类和 java.io.FileOutputStream 类）和通过 JDOM 解析 XML 文件。

JDOM 因其简洁易懂的 API 被广泛地使用。JDOM 提供了 org.jdom、org.jdom.input、org.jdom.output、org.jdom.adapters 和 org.jdom.ransform 包。org.jdom 包中的类是解析 XML 文件所要用到的所有数据类型，org.jdom.input 和 org.jdom.output 两个包分别用来处理 XML 内容的输入、输出。下面介绍 JDOM 解析 XML 文件时用到的类。

- SAXBuilder 类：当要读取 xml 资源时，要使用 input 包中的 SAXBuilder 类从输入流构建文档对象。

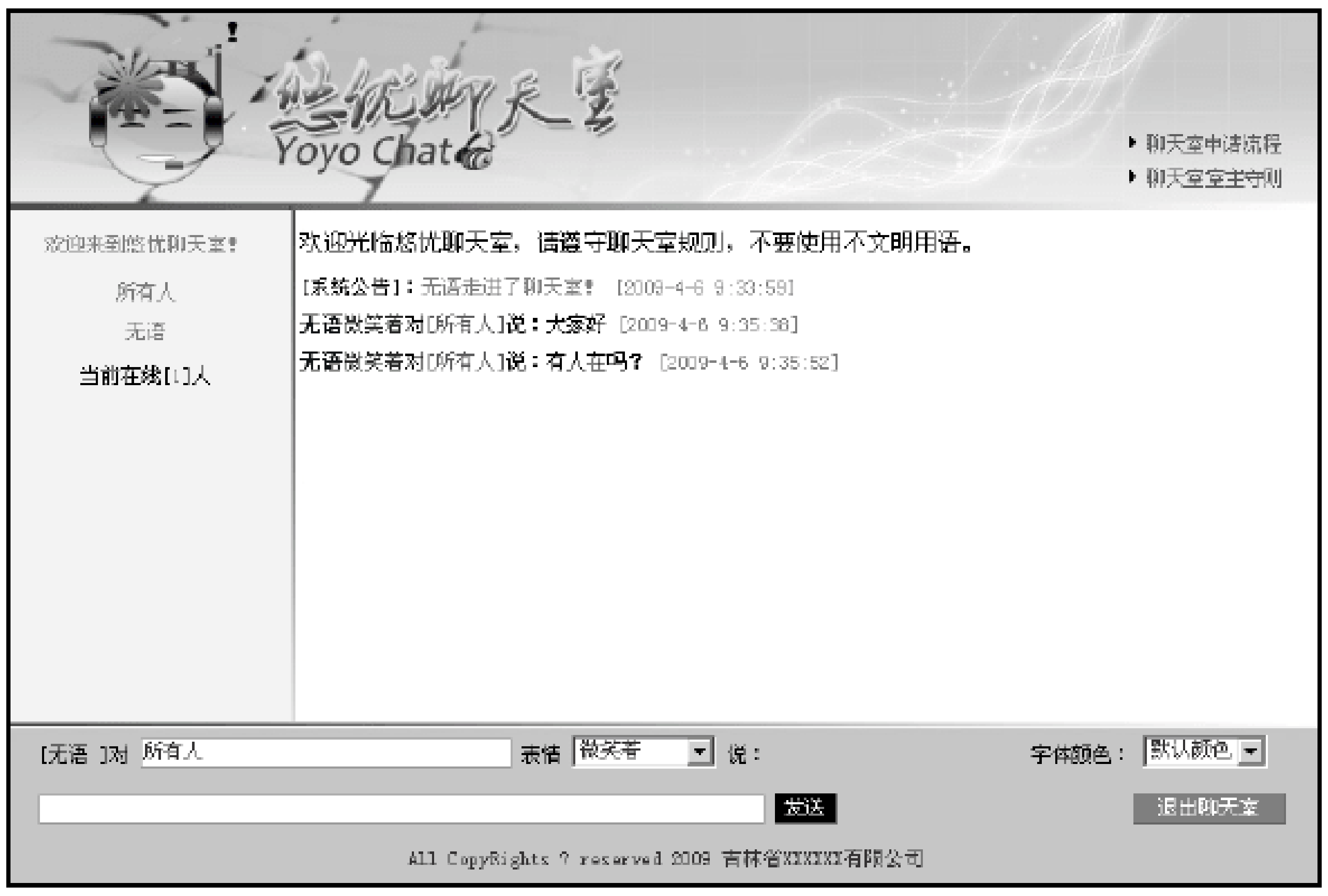


图 23.26　聊天室的主界面

- Document 类：该类代表文档对象。
- Element 类：该类代表 XML 元素。
- Comment 类：该类代表 XML 文件中的注释。

例如， XML 文件中的内容如下：

```
<?xml version="1.0" encoding="GBK"?>
<bookroom>
  <book>
      <bookname>
          Java 数据库系统开发案例精选
      </bookname>
      <author>
            王国辉等
      </author>
      <date>
        2007 年 3 月
      </date>
      <price>
            49 元
      </price>
  </book>
</bookroom>
```

上述文档用 Document 来抽象，bookroom 是文档的根元素，用 Element 类封装，可以通过 Document 对象的 getRootElement()方法获取 Element 对象。Element 元素可以有子元素，bookname、author、date、price 都是子元素，可以通过 Element 对象的 getChildren()方法获取这些子元素。“王国辉等”文本代表了 XML 中的文本值，如元素属性值、元素值、注释的内容等。

设计过程

（1）实现登录聊天室。

编写聊天室登录页面 index.jsp。该页面主要用于收集用户输入的登录信息，以及通过自定义的 JavaScript 函数验证输入信息是否为空和是否包括非法字符。

编写用来保存在线用户和对在线用户进行具体操作的类 UserInfo 和 UserListener。这两个类的代码和实例 597 相同，这里不再赘述。

编写聊天室的 Servlet 实现类，并在该类中添加登录聊天室的方法 loginRoom()，在该方法中，首先获取登录用户，然后判断该用户是否登录。如果已经登录，给出提示信息，否则将该用户添加到在线用户列表中，并且向 Application 对象的聊天内容属性中添加一条系统公告信息，最后将页面重定向到登录成功页。loginRoom()方法的具体代码如下：

```
public void loginRoom(HttpServletRequest request, HttpServletResponse response) {
 response.setContentType("text/html;charset=GBK");
```

```
HttpSession session = request.getSession();
StringUtils su=new StringUtils();
String username=su.toGBK(request.getParameter("username"));            //获得登录用户名
UserInfo user=UserInfo.getInstance();                                  //获得 UserInfo 类的对象
session.setMaxInactiveInterval(600);                                   //设置 session 的过期时间为 10 分钟
Vector vector=user.getList();
boolean flag=true;                                                     //标记是否登录的变量
//判断用户是否登录
if(vector!=null&&vector.size()>0){
	for(int i=0;i<vector.size();i++){
		if(user.equals(vector.elementAt(i))){
			PrintWriter out;
			try {
				out = response.getWriter();
		out.println("<script language='javascript'>alert('该用户已经登录');window.location.href='index.jsp';</script>");
			} catch (IOException e) {
				e.printStackTrace();
			}
			flag=false;
			break;
		}
	}
}
//保存用户信息
if(flag){
	UserListener ul=new UserListener();                                //创建 UserListener 的对象
	ul.setUser(username);                                              //添加用户
	user.addUser(ul.getUser());                                        //添加用户到 UserInfo 类的对象中
	session.setAttribute("user",ul);                                   //将 UserListener 对象绑定到 session 中
	session.setAttribute("username",username);                         //保存当前登录的用户名
	session.setAttribute("loginTime",new Date().toLocaleString());     //保存登录时间
}
/** ******************开始系统公告********************************* */
String fileURL = createFile(request, response);                        //当文件不存在时创建该文件
//获取当前用户
SAXBuilder builder = new SAXBuilder();
try {
	Document feedDoc = builder.build(fileURL);
Element root = feedDoc.getRootElement();
Element channel = root.getChild("messages");
Element newNode = new Element("message");
channel.addContent(newNode);                                           //创建 messages 节点
Element fromNode = new Element("from").setText("[系统公告]");
newNode.addContent(fromNode);
Element faceNode = new Element("face").setText("");
newNode.addContent(faceNode);
Element toNode = new Element("to").setText("");
newNode.addContent(toNode);
Element contentNode = new Element("content").setText("<font color='gray'>" + username + "走进了聊天室！ </font>");
newNode.addContent(contentNode);
//登录时间
Element sendTimeNode = new Element("sendTime").setText(new Date().toLocaleString());
newNode.addContent(sendTimeNode);
request.getRequestDispatcher("login_ok.jsp").forward(request, response);
XMLOutputter xml = new XMLOutputter(Format.getPrettyFormat());
xml.output(feedDoc, new FileOutputStream(fileURL));
} catch (Exception e) {
	e.printStackTrace();
}
}
```

编写登录成功页 login_ok.jsp，在该文件中将页面重定向到聊天室的主界面。登录成功页的具体代码如下：

```
<%@ page contentType="text/html;charset=GBK" language="java" %>
<%	response.sendRedirect("main.jsp"); %>
```

（2）编写聊天室主界面 main.jsp。该页面共包括页面头部、聊天内容显示区、在线人员列表区和用户发言区 4 部分，具体布局如图 23.27 所示。

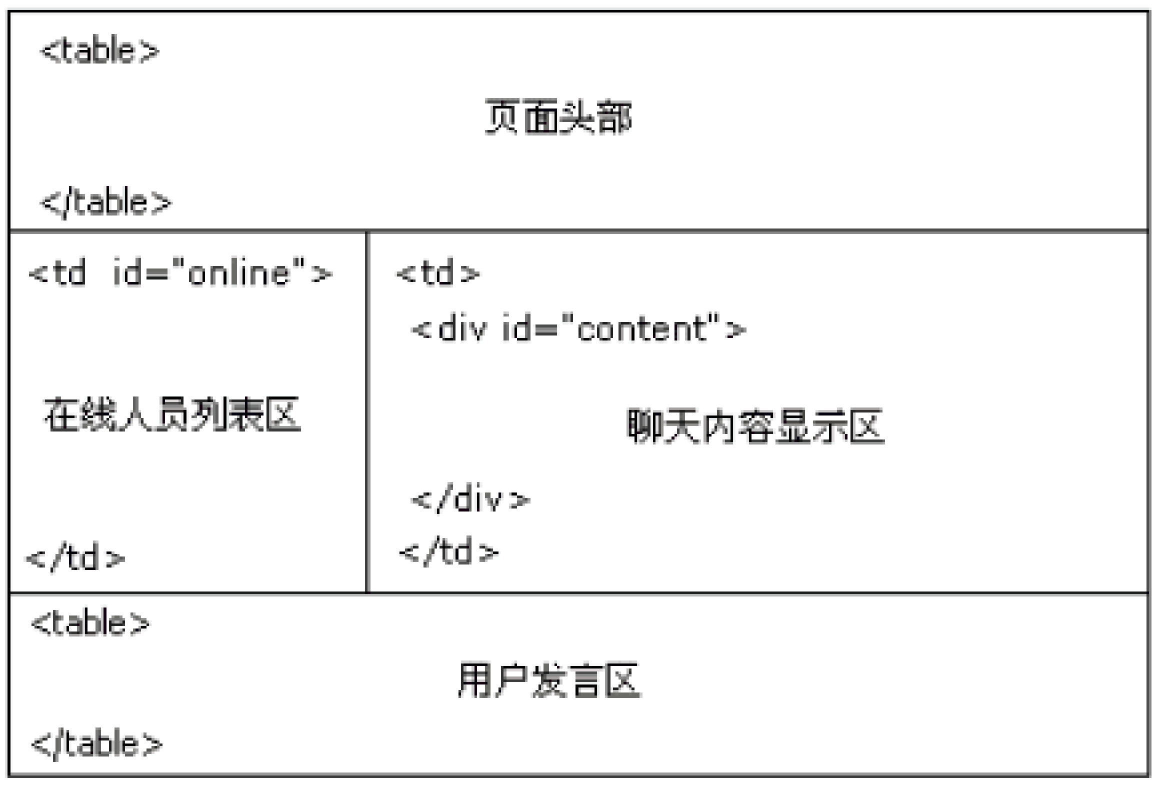


图 23.27　主界面的布局

布局后的主界面的关键代码如下：

```
……          <!--此处省略了页面头部的 HTML 代码-->
<table width="779" height="276" border="0" align="center" cellpadding="0" cellspacing="0">
  <tr>
    <td width="613"   height="200px" valign="top" bgcolor="#FFFFFF"
      style="padding:5px; background-image:url(images/right.jpg); background-repeat:no-repeat;">
<div style="height:290px; overflow:hidden" id="content">聊天内容</div>
</td>
    <td width="165" valign="top" bgcolor="#F3F3F3" id="online" style="padding:5px">
在线人员列表
</td>
  </tr>
</table>
<table width="779" height="64" border="0" align="center" cellpadding="0" cellspacing="0" background="images/bottom.jpg">
……          <!--此处省略了用户发言区的代码，该代码将在实现用户发言时进行详细介绍-->
</table>
```

（3）实现用户发言。

在页面的合适位置添加用于收集用户发言信息的表单及表单元素，关键代码如下：

```
<form action="" name="form1" method="post" >
    <input name="from" type="hidden" value="<%=session.getAttribute("username")%>">[<%=session.getAttribute("username")%> ]对
      <input name="to" type="text" value="" size="35" readonly="readonly">
表情
<select name="face" class="wenbenkuang">
  <option   value="无表情的">无表情的</option>
  <option value="微笑着" selected>微笑着</option>
 <!--此处省略了添加其他列表项的代码-->
  <option value="无精打采的">无精打采的</option>
</select>
说：
    字体颜色：
      <select name="color" size="1" class="wenbenkuang" id="select">
        <option selected>默认颜色</option>
        <option style="color:#FF0000" value="FF0000">红色热情</option>
        <!--此处省略了添加其他列表项的代码-->
        <option style="color:#999999" value="999999">烟雨蒙蒙</option>
      </select>
    <input name="content1" type="text" size="70" onKeyDown="if(event.keyCode==13 && event.ctrlKey){send();}">
      <input name="Submit2" type="button" class="btn_blank" value="发送" onClick="send()">
<input name="button_exit" type="button" class="btn_orange" value="退出聊天室" onClick="Exit()">
</form>
```

编写自定义的 JavaScript 函数 send()，用于调用 Ajax 实现用户发言。在该函数中，首先验证用户输入信息的合法性，然后再将提交的表单元素的内容连接为一个参数字符串，并实例化 Ajax 对象。send()函数的具体代码如下：

```
function send(){     //验证聊天信息并发送
 if(form1.to.value==""){
      alert("请选择聊天对象！");return false;
 }
 if(form1.content1.value==""){
      alert("发送信息不可以为空！");form1.content1.focus();return false;
```

```
}
var param="from="+form1.from.value+"&face="+form1.face.value+"&color="+form1.color.value+"&to="+
form1.to.value+"&content="+      form1.content1.value;
var loader=new net.AjaxRequest("MessagesAction?action=sendMessage",deal_send,onerror,"POST",param);
}
```

在聊天室相关的 Servlet 实现类中，添加发送聊天信息的方法 sendMessages()。在该方法中，首先获取用户发言的相关信息，并对可能出现中文的信息进行转码，然后判断保存当天聊天信息的 XML 文件是否存在，如果不存在，则创建该 XML 文件，最后将聊天信息保存到 XML 文件中，并重定向网页。sendMessages()方法的具体代码如下：

```
public void sendMessages(HttpServletRequest request, HttpServletResponse response){
response.setContentType("text/html;charset=GBK");
StringUtils su = new StringUtils();
Random random = new Random();
String from = su.toUTF8(request.getParameter("from"));                //发言人
String face = su.toUTF8(request.getParameter("face"));                //表情
String to = su.toUTF8(request.getParameter("to"));                    //接收者
String color = request.getParameter("color");                         //字体颜色
String content = su.toUTF8(request.getParameter("content"));          //发言内容
String sendTime = new Date().toLocaleString();                        //发言时间
/** ******************开始添加聊天信息********************************* */
String fileURL = createFile(request, response);                       //当文件不存在时创建该文件
SAXBuilder builder = new SAXBuilder();
Document feedDoc;
try {
    feedDoc = builder.build(fileURL);
    Element root = feedDoc.getRootElement();
    Element channel = root.getChild("messages");
    Element newNode = new Element("message");
    channel.addContent(newNode);                                      //创建 messages 节点
    Element fromNode = new Element("from").setText(from);
    newNode.addContent(fromNode);                                     //添加发言人子节点
    Element faceNode = new Element("face").setText(face);
    newNode.addContent(faceNode);                                     //添加表情子节点
    Element toNode = new Element("to").setText(to);
    newNode.addContent(toNode);                                       //添加接收者子节点
    Element contentNode = new Element("content").setText("<font color='"+ color + "'>" + content + "</font>");
    newNode.addContent(contentNode);                                  //添加聊天内容子节点
    //发言时间
    Element sendTimeNode = new Element("sendTime").setText(sendTime);
    newNode.addContent(sendTimeNode);                                 //添加发言时间子节点
    request.getRequestDispatcher(
            "MessagesAction?action=getMessages&nocache="+ random.nextInt(10000)).forward(request, response);
    XMLOutputter xml = new XMLOutputter(Format.getPrettyFormat());
    xml.output(feedDoc, new FileOutputStream(fileURL));
    /** ****************添加结束****************************** */
} catch (Exception e) {
    e.printStackTrace();
}
}
```

在上面的代码中调用了 createFile()方法，该方法用于根据当前日期生成 XML 文件名，并判断该文件是否存在，如果不存在将创建该文件，具体代码如下：

```
public String createFile(HttpServletRequest request, HttpServletResponse response) {
Date date = new Date();
String newTime = new SimpleDateFormat("yyyyMMdd").format(date);
String fileURL = request.getRealPath("xml/" + newTime + ".xml");
/** **************判断 XML 文件是否存在，如果不存在则创建该文件********** */
File file = new File(fileURL);
if (!file.exists()) {                                                 //判断文件是否存在，如果不存在，则创建该文件
    try {
        file.createNewFile();                                         //创建文件
        String dataStr = "<?xml version=\"1.0\" encoding=\"GBK\"?>\r\n";
        dataStr = dataStr + "<chat>\r\n";
        dataStr = dataStr + "<messages></messages>";
        dataStr = dataStr + "</chat>\r\n";
```

```
                byte[] content = dataStr.getBytes();
                FileOutputStream fout = new FileOutputStream(file);
                fout.write(content);                                    //将数据写入输出流
                fout.flush();                                           //刷新缓冲区
                fout.close();                                           //关闭输出流
            } catch (IOException e) {
                e.printStackTrace();
            }
    }
    return fileURL;
}
```

（4）实现实时显示聊天内容。

编写自定义的 JavaScript 函数 showContent()，用于实例化 Ajax 对象。showContent()函数的具体代码如下：

```
function showContent(){
 var loader1=new net.AjaxRequest("Messages?action=getMessages&nocache="+
        new Date().getTime(),deal_content,onerror,"GET");
}
```

在聊天室相关的 Servlet 实现类中，编写 getMessages()方法。在该方法中，首先判断保存当前聊天信息的 XML 文件是否存在。如果不存在，则创建它，然后开始解析保存聊天信息的 XML 文件，并将聊天内容连接为一个字符串。最后将连接后的字符串保存到 HttpServletRequest 对象中，并重定向网页到输入聊天内容的页面 content.jsp。getMessages()方法的具体代码如下：

```
public void getMessages(HttpServletRequest request,HttpServletResponse response) {
 response.setContentType("text/html;charset=GBK");
  String fileURL = createFile(request, response);                       //当文件不存在时创建该文件
/*******************开始解析保存聊天内容的 XML 文件*********************/
try {
        SAXBuilder builder = new SAXBuilder();
        Document feedDoc = builder.build(fileURL);
        Element root = feedDoc.getRootElement();                        //获取根节点
        Element channel = root.getChild("messages");                    //获取 messages 节点
        Iterator items = channel.getChildren("message").iterator();     //获取 message 节点
        String messages = "";
        //获取当前用户
        HttpSession session = request.getSession();
        String userName = "";
        if (null == session.getAttribute("username")) {
                request.setAttribute("messages", "error");              //保存标记信息，表示用户账户已经过期
        } else {
                userName = session.getAttribute("username").toString();
                DateFormat df = DateFormat.getDateTimeInstance();
                while (items.hasNext()) {
                        Element item = (Element) items.next();
                        String sendTime = item.getChildText("sendTime");        //获取发言时间
                        try {
                                if (df.parse(sendTime).after(
                                                df.parse(session.getAttribute("loginTime").toString()))
                                                || sendTime.equals(session.getAttribute("loginTime").toString())) {
                                        String from = item.getChildText("from");        //获取发言人
                                        String face = item.getChildText("face");        //获取表情
                                        String to = item.getChildText("to");            //获取接收者
                                        String content = item.getChildText("content");  //获取发言内容
                                        if ("[系统公告]".equals(from)) {                 //获取系统公告信息
                                                messages += "[系统公告]：" + content+ " <font color='gray'>["
                                                                + sendTime + "]</font><br>";
                                        } else {                                        //获取普通发言信息
                                                messages += "<font color='blue'><b>" + from+ "</b></font><font color='#CC0000'>"
                                                                + face+ "</font>对<font color='green'>[" + to+ "]</font>说：" + content
                                                                + " <font color='gray'>["+ sendTime + "]</font><br>";
                                        }
                                }
                        } catch (Exception e) {
                                System.out.println("" + e.getMessage());
                        }
                }
```

```
                request.setAttribute("messages", messages);                  //保存获取的聊天信息
        }
        request.getRequestDispatcher("content.jsp").forward(request,response);
} catch (Exception e) {
        e.printStackTrace();
}
}
```

编写显示聊天内容的 JSP 页面 content.jsp，在该页面中只需要应用 out 对象的 println()方法将返回的执行结果输出即可，具体代码如下：

```
<%@ page contentType="text/html; charset=gbk" language="java" import="java.util.*" errorPage="" %>
<% request.setCharacterEncoding("gbk"); %>
<%out.println(request.getAttribute("messages").toString());
%>
```

在聊天室的主界面中，在右侧用于显示聊天内容的单元格，添加一个 id 属性为 content 的<div>标记，用于实时显示聊天内容，具体代码如下：

```
<div style="height:290px; overflow:hidden" id="content">聊天内容</div>
```

编写 Ajax 的回调函数 deal_content()，在该函数中，首先获取 Ajax 处理页的返回值，然后去除字符串中的 Unicode 空白符，最后判断在获取信息时是否产生错误，如果是，则退出聊天室，否则将获取的聊天内容赋值给 id 为 content 的<div>标记的 innerHTML 属性。deal_content()函数的具体代码如下：

```
function deal_content(){
 var returnValue=this.req.responseText;                    //获取 Ajax 处理页的返回值
 var h=returnValue.replace(/\s/g,"");                      //去除字符串中的 Unicode 空白符
 if(h=="error"){
        Exit();
 }else{
        content.innerHTML=sysBBS+returnValue+"</span>";
        //当聊天信息超过一屏时，设置最先发送的聊天信息不显示
        document.getElementById('content').scrollTop = document.getElementById('content').scrollHeight*2;
 }
}
```

为了让页面载入后就调用 Ajax 获取聊天内容，并且每隔 1 秒钟便获取一次数据，还需要在页面中添加以下 JavaScript 代码：

```
window.setInterval("showContent();",1000);
window.onload=function(){
 showContent();                                            //当页面载入后显示聊天内容
}
```

（5）实现退出聊天室。本实例中实现退出聊天室的方法和实例 597 相同，这里不再赘述。

秘笈心法

本实例主要是应用 JDOM 组件将用户的聊天内容保存到 XML 文件中，然后在聊天室中显示聊天内容时，再应用 JDOM 读取保存在 XML 文件中的聊天内容。

23.6　万　年　历

万年历可以显示很多年以前或很多年以后的日期、星期数等，这对于日常生活来说是很方便的。下面将通过几个实例介绍如何制作简易万年历及带阴历的万年历。

实例 599	简易万年历 光盘位置：光盘\MR\23\599	高级 实用指数：★★★

实例说明

在开发某些网站或系统时，可以借用万年历实现填写日期或日期查询的功能。本实例是一个万年历的程序，

可以显示所有的日期及日期所对应的星期数；当前日期所在的表格显示为灰蓝色（图 23.28 所示中 28 日为当前日期）；可以通过按钮查看上个月和下个月，单击“上月”按钮时显示上个月的所有日期，单击“下月”按钮则显示下个月的所有日期，如图 23.28 所示。



图 23.28　简易万年历

关键技术

本实例应用 Calendar 对象获取所需要的时间参数，它是一个抽象类，位于 java.util.Calendar 包，其派生的一个子类为 java.util.GregorianCalendar。

如果想设置、获取或操纵一个日期对象的各个特定部分，如获得小时、日、分钟或计算一个月的某一天是星期几等，就需要使用抽象类 java.util.Calendar 和它的子类来处理。具体的使用方法将在实现过程中介绍。

设计过程

在图 23.28 中显示的是当前的日期，单击“上月”按钮后触发本页面，通过传递参数显示 2 月的日历；单击“下月”按钮后触发本页面，通过传递参数显示 4 月的日历，其关键代码如下：

```
<%@ page language="java" import="java.util.*" %>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>简易万年历</title>
<link href="css/style.css" type="text/css" rel="stylesheet">
</head>
<%! String days[]; %>
<body>
<%
days=new String[42];
for(int i=0;i<42;i++)
{
days[i]="";
}
%>
<%
GregorianCalendar currentDay = new GregorianCalendar();
int today=currentDay.get(Calendar.DAY_OF_MONTH);                    //取得当前日期
int month=currentDay.get(Calendar.MONTH);                           //取得当前月份
int year= currentDay.get(Calendar.YEAR);                            //取得当前年份
//通过 Request 取值，不为空时，程序运行如下代码
if(request.getParameter("month")!=null&&request.getParameter("year")!=null){
int requestMonth=Integer.parseInt(request.getParameter("month"));
int requestYear=Integer.parseInt(request.getParameter("year"));
if(requestYear==currentDay.get(Calendar.YEAR)&&requestMonth==month)
{}                          //当通过 Request 对象取得年和月的数据与当前的年和月的数据相等时，不执行以下代码
else if(requestMonth==-1){  //当通过 Request 对象取得月为-1 时，执行以下代码的赋值
month=11;
year=requestYear-1;
}else if(requestMonth==12){ //当通过 Request 对象取得月为 12 时，执行以下代码的赋值
month=0;
year=requestYear+1;
} else{                     //否则执行以下代码的赋值方法
month=requestMonth;
year=requestYear;
}
}
Calendar thisMonth=Calendar.getInstance();
thisMonth.set(Calendar.MONTH, month );
thisMonth.set(Calendar.YEAR, year );
thisMonth.setFirstDayOfWeek(Calendar.SUNDAY);
thisMonth.set(Calendar.DAY_OF_MONTH,1);
int firstIndex=thisMonth.get(Calendar.DAY_OF_WEEK)-1;               //月份的 1 日为星期几
int maxIndex=thisMonth.getActualMaximum(Calendar.DAY_OF_MONTH);     //月份为多少天
```

```
for(int i=0;i<maxIndex;i++)
{
days[firstIndex+i]=String.valueOf(i+1);
}
%>
<body>
<table width="294" height="32" border="0" align="center" cellpadding="0" cellspacing="0">
  <tr align="center">
    <td background="Image/board_left.gif"><b><font color="#FFFFFF"><%=year%>年<%=month+1%>月</font></b></td>
  </tr>
</table>
<table width="294" height="81" border="0" align="center">
<div align=center>
<tr>
<td   height="16">周日</td><td>周一</td><td>周二</td><td>周三</td><td>周四</td><td>周五</td><td>周六</td>
</tr>
<% for(int j=0;j<6;j++) { %>                                    //列循环
<tr>
<% for(int i=j*7;i<(j+1)*7;i++) {                               //行循环
%>
<%if((i-firstIndex+1)==today){                                  //当前日期为显示的日期时，执行以下代码
%>
<td width="27" height="16"   align="center" bgcolor="#9CA6C6"><font color="#FFFFFF"><b><%=days[i]%></b></font></td>
<%}else {%>
<td width="27" height="16"   align="center"><%=days[i]%></td>
<%} }%></tr><% } %></div>
</table>
<table width="294" height="0" border="0" align="center" cellpadding="0" cellspacing="0">
  <tr align="center">
    <td width="98" height="22"><input type="button" name="Submit1" value="上月" onClick="javascript:window.location.href= 'index.jsp?month=
<%=month-1%>&year=<%=year%>'"></td>
      <td width="98"> <%if(year!=currentDay.get(Calendar.YEAR)||month!=currentDay.get(Calendar.MONTH)){%><input type="button" name="Submit2"
value="返回当前月" onClick="javascript:window.location.href='index.jsp'"> <%}%> </td>
      <td width="98">
        <input type="button" name="Submit3" value="下月" onClick="javascript:window.location.href='index.jsp?month=<%= month+1%>&year=
<%=year%>'">
    </td> </tr>
</table>
</body>
</html>
```

秘笈心法

java.util.Calendar 类就是一个用于表示日历的类，该类提供了用于获取有关日历的所有信息，如年、月、日、星期等。

实例 600	带阴历的万年历 光盘位置：光盘\MR\23\600	高级 实用指数：★★★

实例说明

实例 599 实现的万年历可以满足用户查询公历日期的要求，但是不能满足用户查询阴历日期的需要。本实例实现的是带有阴历的万年历，并且还提供了“上一年”“下一年”“上一月”“下一月”超链接，运行结果如图 23.29 所示。

关键技术

在万年历的程序中，不管是获取农历日期还是公历日期都离不开时间类的支持。下面向读者介绍 Java 中处理时间日期的类 Calendar。

Calendar 类是一个抽象类，该类为特定的瞬间（指的就是某一特定的时刻）与一组诸如 YEAR、MONTH、

DAY_OF_MONTH、HOUR 等日历之间的转换提供了方法，并提供了操作日历字段的常用方法。

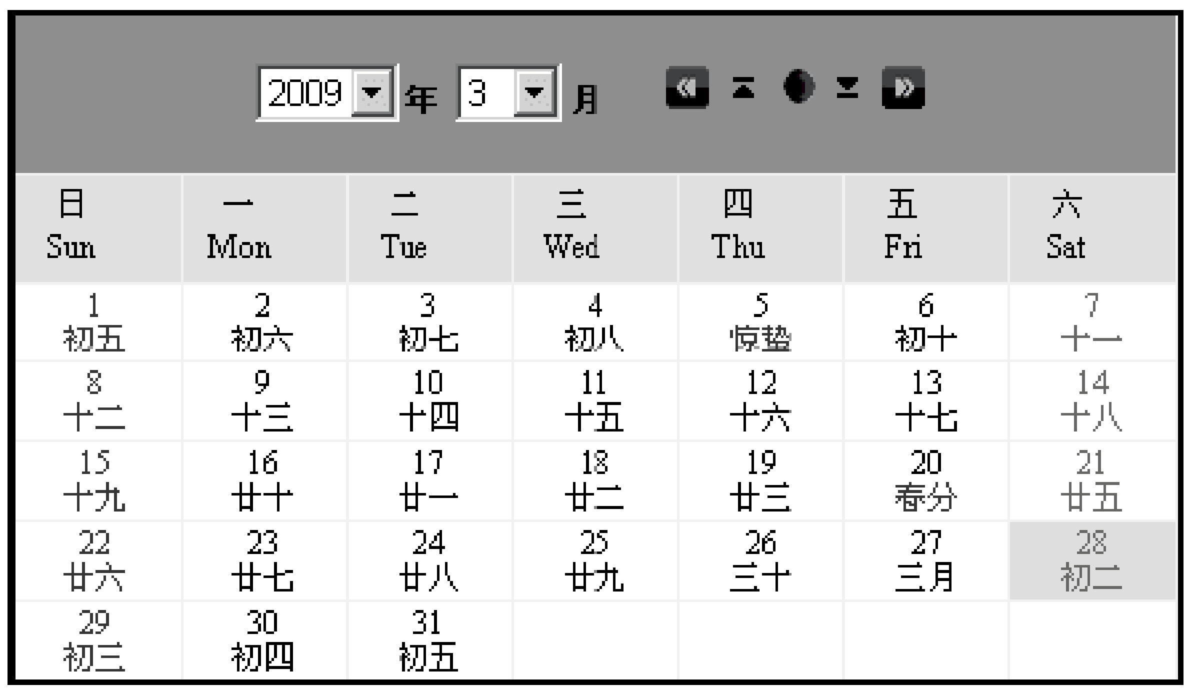


图 23.29　带阴历的万年历

❑ getInstance()方法

该方法是类方法，以获得此类型的一个通用的对象。getInstance()方法返回一个 Calendar 对象。可以通过如下语句获取 Calendar 对象：

```
Calendar today1 = Calendar.getInstance();
```

❑ getTime()方法

该方法可获取表示此 Calendar 时间值的 Date 对象。获取当前系统时间可通过如下语句实现：

```
Calendar rightNow = Calendar.getInstance();
Date date = rightNow.getTime();
```

❑ setTime()方法

该方法参数为 Date 类型对象，通过该对象设置 Calendar 的时间。

设计过程

（1）创建类 MyCalendar。该类包含生成万年历时需要的核心方法，如获取农历 Y 年的总天数、获取 Y 年第 n 个节气为几日等方法。在该类中定义全局变量，关键代码如下：

```
private int year;                                                                   //历年
private int month;                                                                  //历月
private int day;                                                                    //历日
private boolean leap;                                                               //闰年
final static String chineseNumber[] = { "一", "二", "三", "四", "五", "六", "七","八", "九", "十","十一","十二"};   //中文数字
static SimpleDateFormat chineseDateFormat = new SimpleDateFormat("yyyy 年 MM 月 dd 日");   //日期格式
final static long[] lunarInfo = new long[] { 0x04bd8, 0x04ae0, 0x0a570,
 0x054d5, 0x0d260, 0x0d950, 0x16554, 0x056a0, 0x09ad0, 0x055d2,
 0x04ae0, 0x0a5b6, 0x0a4d0, 0x0d250, 0x1d255, 0x0b540, 0x0d6a0,
 0x0ada2, 0x095b0, 0x14977, 0x04970, 0x0a4b0, 0x0b4b5, 0x06a50,
 0x06d40, 0x1ab54, 0x02b60, 0x09570, 0x052f2, 0x04970, 0x06566,
 0x0d4a0, 0x0ea50, 0x06e95, 0x05ad0, 0x02b60, 0x186e3, 0x092e0,
 0x1c8d7, 0x0c950, 0x0d4a0, 0x1d8a6, 0x0b550, 0x056a0, 0x1a5b4,
 0x025d0, 0x092d0, 0x0d2b2, 0x0a950, 0x0b557, 0x06ca0, 0x0b550,
 0x15355, 0x04da0, 0x0a5d0, 0x14573, 0x052d0, 0x0a9a8, 0x0e950,
 0x06aa0, 0x0aea6, 0x0ab50, 0x04b60, 0x0aae4, 0x0a570, 0x05260,
 0x0f263, 0x0d950, 0x05b57, 0x056a0, 0x096d0, 0x04dd5, 0x04ad0,
 0x0a4d0, 0x0d4d4, 0x0d250, 0x0d558, 0x0b540, 0x0b5a0, 0x195a6,
 0x095b0, 0x049b0, 0x0a974, 0x0a4b0, 0x0b27a, 0x06a50, 0x06d40,
 0x0af46, 0x0ab60, 0x09570, 0x04af5, 0x04970, 0x064b0, 0x074a3,
 0x0ea50, 0x06b58, 0x055c0, 0x0ab60, 0x096d5, 0x092e0, 0x0c960,
 0x0d954, 0x0d4a0, 0x0da50, 0x07552, 0x056a0, 0x0abb7, 0x025d0,
 0x092d0, 0x0cab5, 0x0a950, 0x0b4a0, 0x0baa4, 0x0ad50, 0x055d9,
 0x04ba0, 0x0a5b0, 0x15176, 0x052b0, 0x0a930, 0x07954, 0x06aa0,
 0x0ad50, 0x05b52, 0x04b60, 0x0a6e6, 0x0a4e0, 0x0d260, 0x0ea65,
 0x0d530, 0x05aa0, 0x076a3, 0x096d0, 0x04bd7, 0x04ad0, 0x0a4d0,
 0x1d0b6, 0x0d250, 0x0d520, 0x0dd45, 0x0b5a0, 0x056d0, 0x055b2,
 0x049b0, 0x0a577, 0x0a4b0, 0x0aa50, 0x1b255, 0x06d20, 0x0ada0};                //农历日期
String[] solarTerm = {"小寒","大寒","立春","雨水","惊蛰","春分","清明","谷雨",
        "立夏","小满","芒种","夏至","小暑","大暑","立秋","处暑","白露","秋分",
        "寒露","霜降","立冬","小雪","大雪","冬至"};                                   //节气的中文名称
int[] sTermInfo ={0,21208,42467,63836,85337,107014,128867,150921,173149,195551,
```

```
        218072,240693,263343,285989,308563,331033,353350,375494,397447,419210,
        440795,462224,483532,504758};                              //各节气与小寒之间的时间差，单位为分钟
```

（2）定义返回某年的第 n 个节气为几日的方法 sTerm()，该方法包含两个 int 型参数，分别代表农历年 Y 和第 n 个节气，返回值为具体节气所在的日期，具体代码如下：

```
public int sTerm(int y,int n) {
    Calendar offDate = Calendar.getInstance();
    offDate.set(1900, 0, 6, 2, 5, 0);                            //设置基准日期为 1900 年 1 月 6 日 2:05:00 AM
    long temp = offDate.getTime().getTime();
    offDate.setTime(new Date((long) ((31556925974.7 * (y - 1900) + sTermInfo[n] * 60000L) + temp)));
    return offDate.get(Calendar.DAY_OF_MONTH);                   //将具体的天数返回
}
```

（3）创建获取农历 Y 年的总天数方法 yearDays()，该方法有一个 int 型参数，用于指定年份，返回值为农历 Y 年的总天数，具体代码如下：

```
final private static int yearDays(int y) {
    int i, sum = 348;                                            //sum 表示没有闰月时农历年的天数
    for (i = 0x8000; i > 0x8; i >>= 1) {
        if ((lunarInfo[y - 1900] & i) != 0)
            sum += 1;
    }
    return (sum + leapDays(y));
}
```

（4）编写获取农历 Y 年哪个月是闰月的方法 leapMonth()，该方法有一个 int 型参数，用于指定年份，如果该年中没有闰月则返回 0，否则将返回闰月的月份，具体代码如下：

```
final private static int leapMonth(int y) {
    return (int) (lunarInfo[y - 1900] & 0xf);
}
```

（5）编写获取农历 Y 年 m 月的总天数方法 monthDays()，该方法有两个 int 型参数，分别表示农历年 Y 和月 m。返回值为农历 Y 年 m 月的总天数，具体代码如下：

```
final private static int monthDays(int y, int m) {
 if ((lunarInfo[y - 1900] & (0x10000 >> m)) == 0)
     return 29;
 else
     return 30;
}
```

（6）编写转换农历日期中中文日期的方法 getChinaDayString()，该方法有一个 int 型参数，用于指定农历的历日，返回值为中文日期，具体代码如下：

```
public static String getChinaDayString(int day) {
 String chineseTen[] = { "初", "十", "廿", "三" };              //定义中文农历日期需要的中文日期数组
 int n = day % 10 == 0 ? 9 : day % 10 - 1;
 if (day > 30)                                                //如果指定的参数大于 30
     return "";                                               //返回 null
 if (day == 10)                                               //参数等于 10
     return "初十";                                           //返回初十
 else
     return chineseTen[day / 10] + chineseNumber[n];          //返回对应的中文日期
}
```

（7）编写转换农历日期中的中文日期的方法 myCalendar()，该方法只用一个 Calendar 类型参数，用于指定农历的历日，返回值为中文日期，具体代码如下：

```
public String myCalendar(Calendar cal) {
    int yearCyl, monCyl, dayCyl;                              //定义 int 型变量，分别代表历年、历月、历日
    int leapMonth = 0;
    Date baseDate = null;                                     //创建 Date 类型变量
    try {
        baseDate = chineseDateFormat.parse("1900 年 1 月 31 日");  //将 Date 类型变量进行初始化
    } catch (ParseException e) {
        e.printStackTrace();
    }
    //求出和 1900 年 1 月 31 日相差的天数
    int offset = (int) ((cal.getTime().getTime() - baseDate.getTime()) / 86400000L);
    dayCyl = offset + 40;
    monCyl = 14;
```

用 offset 减去每农历年的天数计算当天是农历第几天，其中，i 的最终值为农历的年份，offset 是当年的第几天，具体代码如下：

```
int iYear, daysOfYear = 0;
for (iYear = 1900; iYear < 2050 && offset > 0; iYear++) {
    daysOfYear = yearDays(iYear);
    offset -= daysOfYear;
    monCyl += 12;
}
if (offset < 0) {
    offset += daysOfYear;
    iYear--;
    monCyl -= 12;
}
```

计算农历年份，具体代码如下：

```
year = iYear;
yearCyl = iYear - 1864;
leapMonth = leapMonth(iYear);                    //闰哪个月，1～12
leap = false;
                                                 //用当年的天数 offset，逐个减去每月（农历）的天数，求出当天是本月的第几天
int iMonth, daysOfMonth = 0;
for (iMonth = 1; iMonth < 13 && offset > 0; iMonth++) {
    //闰月
    if (leapMonth > 0 && iMonth == (leapMonth + 1) && !leap) {
        --iMonth;
        leap = true;
        daysOfMonth = leapDays(year);
    } else
        daysOfMonth = monthDays(year, iMonth);

    offset -= daysOfMonth;
    //解除闰月
    if (leap && iMonth == (leapMonth + 1))
        leap = false;
    if (!leap)
        monCyl++;
}
```

offset 为 0 且刚才计算的月份是闰月时，要校正，具体代码如下：

```
if (offset == 0 && leapMonth > 0 && iMonth == leapMonth + 1) {
    if (leap) {
        leap = false;
    } else {
        leap = true;
        --iMonth;
        --monCyl;
    }
}
```

当 offset 小于 0 时，也需要校正，具体代码如下：

```
if (offset < 0) {
    offset += daysOfMonth;
    --iMonth;
    --monCyl;
}
```

获取节气的具体代码如下：

```
if (d == sTerm(y, (m - 1) * 2))
    solarTerms = solarTerm[(m - 1) * 2];
else if (d == sTerm(y, (m - 1) * 2 + 1))
    solarTerms = solarTerm[(m - 1) * 2 + 1];
String tempStr="";
   if(!"".equals(sFeast)){
       tempStr=sFeast;
   }
   if(!"".equals(lFeast)){
       if("".equals(tempStr)){
           tempStr=lFeast;
       }else{
```

```
                tempStr=tempStr+"/"+lFeast;
            }
        }
        if(!"".equals(solarTerms)){
            if("".equals(tempStr)){
                tempStr=solarTerms;
            }else{
                tempStr=tempStr+"/"+solarTerms;
            }
        }
```

判断是否为闰月的具体代码如下：

```
    if(!"".equals(tempStr)){
        return "<font color='red'>"+tempStr+"</font>";
    }else{
        if(day==1){
            return  (leap ? "闰" : "") + chineseNumber[month - 1] + "月";
        }else{
            return getChinaDayString(day);
        }
    }
}
```

（8）在 JSP 页面中编写显示万年历代码。首先在 index.jsp 页面中通过<jsp:useBean>动作标记在 Request 范围内应用 JavaBean MyCalendar 类，具体代码如下：

```
<jsp:useBean class="com.tools.MyCalendar" id="calendar" scope="request"/>
```

（9）创建一个 42 个元素的一维数组，用于初始化万年历中的单元格，并将该数组的各元素设置为空字符串，具体代码如下：

```
<%
String days[]=new String[42];
for(int i=0;i<42;i++){
 days[i]="";                                                    //将数组中的元素值设置为空
}
%>
```

（10）根据要显示的日历日期信息生成日历中的公历日期，具体代码如下：

```
<%   Calendar today1 = Calendar.getInstance();                  //获取 Calendar 类对象
  SimpleDateFormat chineseDateFormat = new SimpleDateFormat("yyyy-MM-dd");   //指定日期格式为 yyyy-MM-dd
  int month = 0;
  int year = 0;
  GregorianCalendar currentDay = new GregorianCalendar();
  //当参数 year 不为空时，表示通过控制区对日历中显示的日期进行改变
  if (request.getParameter("year") != null && !"".equals(request.getParameter("year"))) {
        month = Integer.parseInt(request.getParameter("month")) - 1;
        year = Integer.parseInt(request.getParameter("year"));
  } else {                                                      //采用默认值
        month = currentDay.get(Calendar.MONTH);                 //获取当前月
        year = currentDay.get(Calendar.YEAR);                   //获取当前年
  }
  int today = currentDay.get(Calendar.DAY_OF_MONTH);            //获取当前日
  Calendar thisMonth = Calendar.getInstance();
  thisMonth.set(Calendar.MONTH, month);
  thisMonth.set(Calendar.YEAR, year);
  thisMonth.setFirstDayOfWeek(Calendar.SUNDAY);
  thisMonth.set(Calendar.DAY_OF_MONTH, 1);
  int firstIndex = thisMonth.get(Calendar.DAY_OF_WEEK) - 1;     //获取显示月历中的第一天所在的星期
  int maxIndex = thisMonth.getActualMaximum(Calendar.DAY_OF_MONTH);   //获取显示月历的总天数
  for (int i = 0; i < maxIndex; i++) {                          //生成显示月历中的公历日期
        days[firstIndex + i] = String.valueOf(i + 1);
  }
%>
```

（11）在页面中添加下拉列表，用户可根据下拉列表值选择 1901—2049 年间的任意年份进行查询，下拉列表值的选中状态为当前年份，具体代码如下：

```
<select name="year" onChange="this.form.submit();">
    <%for (int i = 2049; i >= 1901; i--) {%>                    //循环获取年份
    <option value="<%=i%>" <%if (i == year) {                   //将年份添加到下拉列表中
```

```
                    out.println("selected");                //当前年份是下拉列表的默认值
                    }%>>
            <%out.println(i);%>
        </option>
        <%}%>
</select>
```

（12）在页面中设计下拉列表，用户可根据下拉列表值选择任意月份进行查询，并设置当前月份为下拉列表值的默认选中状态，具体代码如下：

```
<select name="month" onChange="this.form.submit();">
<%
    for (int i = 1; i <= 12; i++) {                     //循环向下拉列表中添加值
%>
<option value="<%=i%>"
    <%if (i == month + 1) {
        out.println("selected");}%>>                    //设置当前月份为默认值
    <%
        out.println(i);
    %>
</option>
<%}%>
</select>
```

（13）在页面的合适位置创建一个 3 行 7 列的表格，用于显示万年历。其中，第一行用于显示控制区，第二行用于显示日历的标头，第 3 行用于显示日期，关键代码如下：

```
<table width="443" border="0" align="left" style="float:left" cellpadding="0" cellspacing="1" bordercolor="#FFFFFF" bgcolor="#F2F1EF">
   <tr bgcolor="FFFCF1">
     <td height="36" colspan="7" align="center" valign="middle" bgcolor="#AC8F67">
……                                  <!--此处省略了显示控制区的代码-->
</td>
    </tr>
   <tr bgcolor="C9B65A">
     <td width="44" height="40" align="center" bgcolor="#EBDFC9" class="word_red">日<br>
       SUN </font></td>
  <td width="44" height="40" align="center" bgcolor="#EBDFC9" class="word_darkGray">一<br>
    MON</td>
  ……              <!--此处省略了显示星期二至星期五的其他单元格的代码-->
  <td width="44" height="40" align="center" bgcolor="#EBDFC9" class="word_blue">六<br>
    SAT</font></td>
    </tr>
<tr bgcolor="FFFCF1">
     <td align="center" bgcolor="<%=bgcolor%>" valign="middle"></td>
    </tr>
 </table>
```

（14）编写循环生成日历中日期代码。在生成日历内容时，需要调用 MyCalendar 类的 myCalendar()方法生成农历日期，具体代码如下：

```
<%
String color = "#000000";
String bgcolor = "FFFFFF";
for (int j = 0; j < 6; j++) {
%>
<tr bgcolor="FFFCF1">
    <%
        for (int i = j * 7; i < (j + 1) * 7; i++) {
                switch ((i + 1) % 7) {
                case 6:
                case 2:
                case 3:
                case 4:
                case 5:
                        color = "#000000";
                        break;
                case 1:
                        color = "#FF0000";
                        break;
                case 0:
                        color = "#1B789D";
```

```
                    break;
                }
                if ((i - firstIndex + 1) == today) {
                    bgcolor = "#C8E3F3";                        //设置当天日期的背景颜色
                } else {
                    bgcolor = "FFFFFF";
                }
    %>
    <td align="center" bgcolor="<%=bgcolor%>" valign="middle" <%if (!"".equals(days[i])) {%>
        <%int d = Integer.parseInt(days[i]);%>
        <%out.println("style=\"cursor:hand;\"    date=" + year
                            + "-" + (month + 1) + "-" + d
                            + "','';return false;\"><font color=\""
                            + color + "\">" + d + "<br>");
                        if (days[i] != "") {
                            String str = year + "-" + (month + 1) + "-" + d;
                            today1.setTime(chineseDateFormat.parse(str));
                            out.println(calendar.myCalendar(today1)); //获取农历日期或是节气日
                        }
                        out.println("</font>");%>
        <%}%>
        </td>
    <%
      }
    %>
</tr>
<%
}
%>
</table>
```

秘笈心法

在本实例中多次用到了 0x，使用该字符作为前缀表示为十六进制数。